Εθνικο Μετσοβιο Πολυτεχνειο
Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων

# Τεχνικές παραλληλοποίησης και βελτιστοποίησης εξομοιωτή βιολογικών νευρικών δικτύων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**Σωτηρίου Γ. Παναγιώτου**

**Επιβλέπων**: Δημήτριος Ι. Σούντρης
Καθηγητής

Σωτήριος Παναγιώτου

Αθήνα, Οκτώβριος 2018

**Εθνικο Μετσοβιο Πολυτεχνειο**

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ
ΣΥΣΤΗΜΑΤΩΝ

# Parallelization and optimization techniques on a biological neural net simulator

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**Σωτηρίου Γ. Παναγιώτου**

**Επιβλέπων**: Δημήτριος Ι. Σούντρης
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τη 1η Νοεμβρίου 2018.

| ..... | ..... | ..... |
|---|---|---|
| Δημήτριος Ι. Σούντρης<br>Καθηγητής | Κιαμάλ Ζ. Πεκμεστζή<br>Καθηγητής | Γεώργιος Γκούμας<br>Επίκουρος Καθηγητής |

Αθήνα, Οκτώβριος 2018

.....

**Σωτήριος Γ. Παναγιώτου**
Διπλωματούχος Φοιτητής του Εθνικού Μετσόβιου Πολυτεχνείου

# Περίληψη

Η νευροεπιστήμη είναι το επιστημονικό πεδίο που ασχολείται με τη λειτουργία του νευρικού συστήματος. Η πρόοδος στη νευροεπιστήμη έχει ωφελήσει τους περισσότερους τομείς της Ιατρικής και έχει βρει εφαρμογές σε νευροενεργά φάρμακα, βηματοδότες καρδιάς και διεπαφές εγκεφάλου-μηχανής. Μια σημαντική προσθήκη στην πειραματική μέθοδο για τη νευρολογική έρευνα είναι οι προσομοιώσεις σε υπολογιστή, καθώς προσφέρουν μεγάλη οικονομία σε χρήμα και χρόνο, αυξημένη αποδοτικότητα στα πειράματα και διερευνησιμότητα συνθηκών που δε μπορούν να μελετηθούν πειραματικά.

Η αιχμή της έρευνας στην νευροεπιστήμη είναι η μελέτη της συμπεριφοράς πολύ μεγάλων νευρικών δικτύων. Το όριο στο μέγεθος δικτύου που μπορεί να προσομοιωθεί διαμορφώνεται από την κλίμακα του υπολογιστικού συστήματος που χρησιμοποιείται. Η ανάπτυξη λογισμικού σε συστήματα μεγάλης κλίμακας, όμως, περιλαμβάνει τεχνικά ζητήματα παραλληλοποίησης των υπολογισμών και φορητότητας του λογισμικού. Προκύπτει λοιπόν η ανάγκη για έναν προσομοιωτή νευρικών δικτύων που θα διαχειρίζεται αυτόματα και αποδοτικά τους διαθέσιμους πόρους, θα μπορεί να μεταφερθεί άμεσα σε ποικίλες υπολογιστικές υποδομές, και θα επιτρέπει εύκολο χειρισμό, διαλειτουργικότητα και διερεύνηση νέων υπολογιστικών μοντέλων ώστε να ενδυναμώσει τη νευρολογική έρευνα.

Προς αυτή την κατεύθυνση, η παρούσα εργασία αφορά την ανάπτυξη ενός νέου προσομοιωτή μεγάλων νευρικών δικτύων, που δέχεται περιγραφές νευρικών μοντέλων στη βιολογική μορφή που είναι οικεία στους νευροεπιστήμονες, επιτρέπει τη σύνθεση νευρικών μοντέλων που περιλαμβάνουν νέα φαινόμενα και φυσιολογικές διαδικασίες, και μεταφέρεται σε νέες υπολογιστικές υποδομές εκμεταλλευόμενο αποδοτικά τους υπάρχοντες πόρους, όλα χωρίς ειδική παραμετροποίηση από μηχανικό υπολογιστών. Ο προσομοιωτής έχει επίσης σχεδιαστεί ώστε να μπορεί να χρησιμοποιηθεί σε συνεργασία με υπάρχοντα λογισμικά της ερευνητικής διαδικασίας. Για επιβεβαίωση των χαρακτηριστικών του προσομοιωτή, εκτελέστηκε σε υποδομή υπολογιστικού νέφους πλήθος προσομοιώσεων που εξετάζουν την υπολογιστική επίδοση για ένα εύρος πολυπλοκότητας νευρικών διασυνδέσεων, πολυπλοκότητας νευρικής δομής και διαθέσιμων επεξεργαστικών μονάδων. Επίσης σχεδιάστηκε λογισμικό διεπαφής για σύνδεση του προσομοιωτή της εργασίας με τη γενική πλατφόρμα νευρικών προσομοιώσεων BrainFrame.

## Λέξεις Κλειδιά

Amazon AWS, υπολογιστική υψηλής επίδοσης, containerization, OpenMP, κλιμακωσιμότητα, νευροεπιστήμη, ηλεκτροφυσιολογία, in silico ιατρική, PyNN

# Abstract

Neuroscience is the scientific field studying the functions of nervous systems. Advances in neuroscience have benefited most fields of medicine and have enabled applications such as new neuroactive drugs, heart pacemakers and brain-machine interfaces. In neuroscience research, live experiments are complemented by computer simulations, which deliver major cost and time savings, enhance effectiveness of wet lab experiments and provide the ability to investigate conditions that cannot be reproduced experimentally.

The cutting edge of neuroscience research is concerned with the emergent behaviours of large-scale neural nets. The size and complexity of simulated neural nets is limited by the scale of the computational system in use. Software development targeting large-scale computers, though, entails technical issues regarding parallelization of computations and software portability. Thus, the need arises for a neural net simulator that utilizes available resources automatically and efficiently, is readily portable to various high-performance computational systems, and facilitates system usage, interoperability and exploration of new computational models, in order to boost neuroscience research.

Toward this direction, the present thesis involved development of a new neural net simulator, that supports neural models in the physiological form used by neuroscientists, enables design of neural models that include new model and biological processes, and is portable to different computational infrastructure without custom modifications by computer engineers.The simulator has also been designed to be interoperable with established research software.

In order to confirm the simulator's features, the simulator was deployed on cloud infrastructure and a set of simulation runs was performed, exploring computational performance over a range of neural connectivity complexity, neural structure complexity and available processors. In addition, interface software was designed to plug the simulator into the BrainFrame general neural net simulation platform.

## Keywords

Amazon AWS, high performance computing, containerization, OpenMP, scalability, neuroscience, electrophysiology, in silico medicine, PyNN

# Ευχαριστίες

Η διπλωματική εργασία αυτή πραγματοποιήθηκε στο Εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων, του Εθνικού Μετσόβιου Πολυτεχνείου υπο την επίβλεψη του Καθηγητή Δημητρίου Σόυντρη.

Καταρχήν θα ήθελα να ευχαριστήσω τον Καθ. Δημήτριο Σούντρη για την ευκαιρία που μου έδωσε να δουλέψω στην πρώτη γραμμή της Υπολογιστικής Υψηλών Επιδόσεων και την καθοδήγησή του, καθώς και τον Δρ. Χαράλαμπο Σιδηρόπουλο και τον υποψήφιο διδάκτορα Γεώργιο Χατζηκωνσταντή του Εργαστηρίου, για τις συμβουλές και την υποστήριξή τους, και τον χρόνο που μου αφιέρωσαν κατά τη εκπόνηση της διπλωματικής μου εργασίας, και τον Καθ. Χρήστο Στρύδη, τον υποψήφιο διδάκτορα Γεώργιο Σμάραγδο και τον Επ.Καθ. Mario Negrello του Erasmus Medical Center Rotterdam για τις συμβουλές τους στην προσομοίωση νευρικών κυττάρων, και τον πτυχιακό φοιτητή Rene Miedema του TU Delft για την αρχική έκδοση του προσομοιωτή νευρώνων κάτω ελαίας.

Θα ήθελα ακόμη να ευχαριστήσω τον Δρ. Χριστόφορο Κάχρη για την υποστήριξή του, και τους πόρους Cloud Computing που παραχώρησε για την εκτέλεση των πειραμάτων επίδοσης.

Επίσης, θα ήθελα να ευχαριστήσω τον κ. Κωνσταντίνο Κατσαντώνη για τη βοήθειά του στην ενσωμάτωση του προσομοιωτή στην πλατφόρμα BrainFrame.

Ακόμη, θα ήθελα να ευχαριστήσω από τη Σχολή τον Αναπληρωτή Καθηγητή Νικόλαο Παπασπύρου και τον Επίκουρο Καθηγητή Δημήτριο Φωτάκη για την πολύτιμη διδασκαλία τους στο αντικείμενο των Αλγορίθμων, την Καθηγήτρια Κωνσταντίνα Νικήτα και τον Καθηγητή Γεώργιο Ματσόπουλο που μου καλλιέργησαν το ενδιαφέρον για την Υπολογιστική Βιολογία, τους Καθηγητές και Βοηθούς του Εργαστηρίου Υπολογιστικών Συστημάτων για την διδασκαλία τους για το υλικό και λογισμικό υψηλής επίδοσης, καθώς και τους Δασκάλους που είχα στο σχολείο, που μου κίνησαν και σε μεγάλο βαθμό υποστήριξαν το ενδιαφέρον για τις Θετικές Επιστήμες και την Πληροφορική.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου, για την κατανόηση και την υποστήριξη που μου έχουν προσφέρει όλα αυτά τα χρόνια.

# Contents

# List of Figures

# List of Tables

# Εκτεταμένη Περίληψη

## Εισαγωγή

Η νευροεπιστήμη είναι το επιστημονικό πεδίο που ασχολείται με τη λειτουργία του νευρικού συστήματος των έμβιων οργανισμών. Η πρόοδος στη νευροεπιστήμη έχει συνεισφέρει άμεσα ή έμμεσα στους περισσότερους τομείς της Ιατρικής και έχει βρει εφαρμογές όπως νευροενεργά φάρμακα, βηματοδότες καρδιάς και διεπαφές εγκεφάλου-μηχανής.

Μια σημαντική προσθήκη στην πειραματική μέθοδο για τη νευρολογική έρευνα είναι οι προσομοιώσεις σε υπολογιστή, καθώς έτσι μπορούν να δοκιμαστούν σε πρώτο στάδιο πειράματα, χωρίς τον κόπο, το χρόνο και το κόστος της *in vivo* εφαρμογής, δίνοντας τη σωστή κατεύθυνση για το πού αξίζει να εστιάσουν τα πραγματικά πειράματα. Παράλληλα, οι προσομοιώσεις μπορούν να δώσουν προσεγγιστική εικόνα για φαινόμενα που δε μπορούν να εξεταστούν στο εργαστήριο, όπως επεμβατικά *in vivo* πειράματα χωρίς αναισθησία, εσωκυτταρικές συνθήκες, σπάνιες παθήσεις και ζητούμενες δράσεις για τις οποίες αναζητείται ακόμη μέσο.

Ο κλάδος εφαρμογών υπολογιστών που εκτελούν υπολογιστικές εργασίες σε συστήματα μεγάλης κλίμακας ονομάζεται υπολογιστική μεγάλης κλίμακας.¨

Η υπολογιστική μεγάλης κλίμακας μπορεί να χωριστεί σε δύο επιμέρους τομείς: τον τομέα υπερυπολογιστών, και τον τομέα υπολογιστικού νέφους. Ο τομέας υπερυπολογιστών χρησιμοποιεί συστήματα κορυφαίας επίδοσης αριθμητικών πράξεων, προσανατολισμένα σε επιστημονικές εφαρμογές, συνδεδεμένα μεταξύ τους με ειδικό υλικό διασύνδεσης μεταξύ των κόμβων, με στόχο να εκτελέσουν μεμονωμένες εφαρμογές ακραίας κλίμακας. Ο τομέας υπολογιστικού νέφους χρησιμοποιεί κοινό, εμπορικά παραγόμενο υλικό ώστε να παρέχει υπολογιστική ισχύ ως εμπορευματοποιημένη υπηρεσία σε πλειάδα χρηστών, στοχεύοντας στο χαμηλό κόστος της υπολογιστικής ισχύος και στην τυποποίηση του υπολογιστικού περιβάλλοντος. Λόγω όμως της αυξημένης ζήτησης για υπολογιστική ένταση, έχει αρχίσει να προσφέρεται και υλικό υψηλών επιδόσεων που μπορεί να καλύψει επιστημονικούς χρήστες.

Βάσει των παραπάνω, υπάρχει τρέχουσα προσπάθεια για σύγκλιση των δύο τομέων ώστε να συνδυαστούν τα πλεονεκτήματά τους. Πέρα από τη στροφή των παρόχων υπολογιστικού νέφους που περιγράφηκε παραπάνω, προτείνονται συστήματα-πράκτορες που λαμβάνουν εργασίες προς εκτέλεση από τους χρήστες και τις αναθέτουν σε ετερογενείς παρόχους υπολογιστικής ισχύος, προς βελτιστοποίηση χρόνου εκτέλεσης, εκμετάλλευσης υλικού, ή κατανάλωσης ενέργειας.

# Υπάρχον σχετικό έργο

Όπως αναφέρθηκε παραπάνω, ένα αρχικό στάδιο για τη διερεύνηση των λειτουργιών του νευρικού ιστού είναι ο πειραματισμός σε θεωρητικά μοντέλα. Συνήθως οι ερευνητικές εργασίες αναπτύσσουν επί τούτω λογισμικό προσομοίωσης, εστιάζοντας στο συγκεκριμένο πρόβλημα που απασχολεί τους ερευνητές και τις συγκεκριμένες παραμέτρους που εξετάζουν. Αυτό οδηγεί σε σημαντικό τεχνικό κόπο που αποσπά τους ερευνητές από τους αρχικούς τους στόχους. Τη λύση σε αυτό το πρόβλημα επιχειρούν να δώσουν προσομοιωτές γενικής χρήσης, που έχουν ως στόχο να καλύψουν τα πιο κοινά νευρολογικά χαρακτηριστικά υπό μελέτη και τις πιο κοινές τεχνικές ανάλυσης που εφαρμόζονται.

Καθώς η ταχύτητα σχεδίασης πειραμάτων και προσομοίωσής τους είναι κρίσιμα στον κύκλο της έρευνας, απαιτείται η συνεργασία μεταξύ ειδικών της νευρολογίας και της τεχνολογίας υπολογιστών. Προσφάτως το ενδιαφέρον στην έρευνα έχει στραφεί στα νευρικά δίκτυα μεγάλης κλίμακας, καθώς παρουσιάζουν μη τετριμμένες, ευφυείς λειτουργίες και περιγράφουν τις δομές του εγκεφάλου με μεγαλύτερη ακρίβεια. Η υπάρχουσα τεχνολογία περιορίζει την πολυπλοκότητα των προσομοιούμενων μοντέλων, οπότε έχουν προταθεί καινοτόμα συστήματα υπολογισμού προς αύξηση της πολυπλοκότητας και πιστότητας προσομοίωσης.

Η πολυπλοκότητα των νευρικών δικτύων βρίσκεται στο πλήθος των συνάψεων μεταξύ των νευρώνων, που αντιστοιχούν σε ροές πληροφοριών εντός του δικτύου. Μία προσέγγιση για να καλυφθούν αυτές οι ροές μοιράζει συστάδες νευρώνων σε διάφορους υπολογιστικούς κόμβους. Αυτοί οι κόμβοι επικοινωνούν μεταξύ τους, με τοπολογία εμπνευσμένη από τα φυσικά νευρωνικά δίκτυα. Με αυτό το κατανεμημένο σύστημα αποφεύγεται ο περιορισμός von Neumann. Οι περιορισμοί της τοπολογίας δικτύωσης, όμως, περιορίζουν αντίστοιχα τις ιδιότητες των δυνατών *in silico* δικτύων.

Στις μέρες μας, η υπολογιστική νευροεπιστήμη αναλύει μοντέλα νευρικών δικτύων σε τρία επίπεδα: το επίπεδο όπου οι νευρώνες είναι απλοποιημένα αυτόματα που ανταλλάσουν διακριτά εναύσματα, το επίπεδο όπου τα φυσιολογικά ανατομικά χαρακτηριστικά των κυττάρων λαμβάνονται υπ΄όψιν άμεσα αλλά ως αδρές δομές, και το επίπεδο που προσομοιώνονται τα μικροφυσιολογικά χαρακτηριστικά της μεμβράνης των νευρώνων. Η παρούσα εργασία ασχολείται με την ανάλυση του παραπάνω από αυτά τα επίπεδα.

Στο αδρό φυσιολογικό επίπεδο ανάλυσης, η αλληλεπίδραση μεταξύ νευρώνων συμβαίνει σε συνεχή χρόνο, οπότε αυξάνονται δραματικά οι απαιτήσεις ρυθμού και υστέρησης στην επικοινωνία μεταξύ νευρώνων, σε σχέση με πιο απλά μοντέλα. Υπολογιστικά συστήματα προσομοίωσης σε αυτό το επίπεδο έχουν υλοποιηθεί σε αναλογικούς και ψηφιακούς υπολογιστές.

Ένα αναλογικό ολοκληρωμένο κύκλωμα που εξομοιώνει νευρώνες με κλασικά κανάλια νατρίου και καλίου, με προγραμματίσιμη συνδεσιμότητα μεταξύ των νευρώνων είχε κατασκευαστεί το 2006[FGH06]. Έχει επίσης προταθεί μια γενικευμένη τεχνική εξομοίωσης καναλιών ιόντων από MOSFET προς υλοποίηση δικτύων σε μεγάλη κλίμακα[HB07]. Ακόμη, ο προσομοιωτής BrainScaleS[SFM08] προσομοιώνει δίκτυα με πλήρως αναλογική διασύνδεση, σε επίπεδο μεμονωμένης ψηφίδας. Οι ελλείψεις ευελιξίας του τελικού πυριτίου και η υπερβολική κατανάλωση ενέργειας για επικοινωνία μεταξύ διαφορετικών ψηφίδων, όμως, έχει επιβάλλει τη ψηφιοποίηση ή και διακριτοποίηση της επικοινωνίας σε υλοποιήσεις μεγάλης κλίμακας.

Μια πιο ευέλικτη λύση είναι η ολοκλήρωση των συνήθων διαφορικών εξισώσεων που καθορίζουν αυτά τα μοντέλα σε ψηφιακό υπολογιστή, σε διακριτά χρονικά βήματα. Η προσομοίωση της εσωτερικής φυσιολογίας των κυττάρων χρειάζεται υψηλή χρονική ανάλυση, που αυξάνει ακόμη περισσότερο τη ροή δεδομένων επικοινωνίας μεταξύ των νευρώνων. Πολλές ηλεκτροφυσιολογικές προσομοιώσεις έχουν τρέξει σε γενικής χρήσης υπερυπολογιστές σειράς Blue Gene, με χρήση της παράλληλης εκδοχής του λογισμικού NEURON[Mar06], ή του λογισμικού UC4[Bre+16]. Μια υλοποίηση μειώνει το φόρτο του υλικού με επικοινωνία τιμών τάσης μεταξύ νευρώνων σε χαμηλότερη ανάλυση, και τηλεσκοπική ολοκλήρωση της δυναμικής των νευρώνων σε ηρεμία. Αυτή η υλοποίηση έχει πετύχει καλά αποτελέσματα σε υλικό GPU[Sma+16]. Μια ετερογενής προσέγγιση εξέτασε την υπολογιστική επίδοση επιταχυντών τύπου manycore CPU, GPU, και dataflow engine ως προς παραμέτρους συνδεσιμότητας, επιλέγοντας αυτόματα τη βέλτιστη αρχιτεκτονική, με ενδεικτική εφαρμογή σε μοντέλο κυττάρων κάτω ελαίας του εγκεφάλου.

Μία εναλλακτική προσέγγιση εστιάζει στο μεγαλύτερο δυνατό εύρος υποστηριζόμενων μοντέλων, αντί για το μέγιστο αριθμό νευρώνων ή συνάψεων. Αυτή η προσέγγιση είναι εξαιρετικά χρήσιμη για τους ερευνητές που αναζητούν νέα μοντέλα προσομοίωσης, καθώς δε χρειάζεται να αναπτύξουν λογισμικό από το μηδέν. Ενδεικτικά πακέτα είναι τα NEURON, NEST, BRIAN, MOOSE, και GENESIS. Τα πιο πολλά πακέτα, όμως, έχουν σχεδιαστεί για προσωπικούς υπολογιστές, περιορίζοντας αντίστοιχα την πολυπλοκότητα των μελετήσιμων μοντέλων.

Οι περιορισμοί των προσωπικών υπολογιστών ώθησαν τους ερευνητές να τρέξουν *in silico* πειράματα σε υπερυπολογιστές. Υπάρχουν όμως εμπόδια σε τεχνικό και διαδικαστικό επίπεδο. Για την εκτέλεση του λογισμικού-στόχου σε τέτοια συστήματα, χρειάζεται βοηθητικός αυτοματισμός που χρειάζεται ειδικές γνώσεις, και πρέπει να αναπτυχθεί από την αρχή για κάθε εναλλακτική υποδομή που χρησιμοποιείται. Ακόμη, η διαδικασία παραχώρησης υπολογιστικού χρόνου έχει πολλά στάδια και συχνά περιλαμβάνει επιτροπή έγκρισης για το συγκεκριμένο έργο. Η εφαρμογή των πρακτικών υπολογιστικής νέφους μπορεί όμως να βελτιώσει τη χρηστικότητα και την ευελιξία των συστημάτων υπερυπολογιστών, αφαιρώντας τις εξειδικευμένες τεχνικές λεπτομέρειες.

Μία πλατφόρμα βασισμένη στην υπολογιστική νέφους, που παρέχει όμως υπηρεσίες υψηλής επίδοσης στην υπολογιστική νευροεπιστήμη, είναι η BrainFrame. Ο στόχος της είναι να παρέχει μια εύχρηστη διεπαφή σχεδίασης μοντέλων και πειραμάτων στους ερευνητές, μέσω της οποίας αυτοί θα μπορούν να μπορούν να εκτελέσουν πειράματα σε υλικό υψηλής επίδοσης με πρωτόγνωρη άνεση.

# Περιγραφή προβλήματος

Η υπολογιστική νευροεπιστήμη είναι η μελέτη των υπολογιστικών ιδιοτήτων των νευρικών δικτύων. Αυτή η μελέτη χρησιμοποιεί μοντέλα διαφόρων κλιμάκων λεπτομέρειας, με απώτερο στόχο να δώσει καθολικές απαντήσεις για τη λειτουργία των εγκεφάλων, που είναι οι πιο πολύπλοκες, ικανές και ανεξήγητες νευρικές δομές που υπάρχουν, και των οποίων οι δυνατότητες ξεπερνούν μακράν την υπάρχουσα υπολογιστική τεχνολογία.

Η έρευνα στη νευρολογία προσφέρει διαρκώς νέες φυσιολογικές και ανατομικές πληροφορίες για τον εγκεφαλικό ιστό, μέσω πολλών παράλληλων διεθνών ερευνητικών προγραμμάτων. Για τις κύρια κέντρα του εγκεφάλου, έχουν καταγραφεί οι πηγές και οι προορισμοί από και προς τα οποία διαδίδονται τα διεγερτήρια σήματα. Νέοι φυσιολογικοί μηχανισμοί που επηρεάζουν τη συμπεριφορά των νευρώνων διαρκώς αποκαλύπτονται και αναλύονται, ενώ άλλες προσπάθειες ανακατασκευάζουν τον πλήρη συναπτικό άτλα εγκεφάλων, με χρήση μικροτομών. Καθώς οι ζωντανοί εγκέφαλοι είναι πολύ ευαίσθητα όργανα, τα *in vivo* πειράματα μπορούν να δώσουν μόνο μια πολύ επιφανειακή εικόνα των εσωτερικών διεργασιών που λαμβάνουν μέρος.

Για να έχουν νόημα τα υπολογιστικά μοντέλα, τα αποτελέσματά τους πρέπει να είναι άμεσα συγκρίσιμα με πειραματικά δεδομένα, όπως ηλεκτροεγκεφαλογραφήματα, μετρήσεις εισηγμένων ηλεκτροδίων, και λοιπές τεχνικές πειραμάτων in vitro. Έτσι μπορεί να εκτιμηθεί η ακρίβειά τους και να επισημανθούν τα σημεία που επιδέχονται βελτίωση σε αυτά.

Στο πειραματικό σχεδιασμό δεν είναι πάντα σταθερές όλες οι ελεγχόμενες παράμετροι· κάποιες μπορούν αν μεταβάλλονται ανά πείραμα ώστε να μελετηθεί η διάταξη στο εύρος αυτών των παραμέτρων. Τέτοιες είναι, ενδεικτικά, ο τύπος και η ένταση της εξωτερικής διέγερσης, η παρουσία νευροτρόπων ουσιών, και η πολυπλοκότητα και μυελίνωση των νευρικών δικτύων.

Η παρούσα εργασία ασχολείται με την εφαρμογή της υπολογιστικής νέφους και υψηλών επιδόσεων, στη προσομοίωση σε φυσιολογικό επίπεδο μεγάλων ετερογενών νευρικών δικτύων, με διεπαφή που να μπορεί να χρησιμοποιηθεί από νευροεπιστήμονες χωρίς ειδική εκμάθηση. Μια τέτοια εφαρμογή πρέπει να ανταποκρίνεται στις ανάγκες των χρηστών, της εφαρμοζόμενης τεχνολογίας, και των εργαλείων με τα οποία πρέπει να συνεργάζεται.

Οι νευροεπιστήμονες μελετούν μοντέλα του πώς τα νευρικά κύτταρα λειτουργούν και αλληλεπιδρούν, ώστε να περιγράψουν τη λειτουργία του νευρικού συστήματος. Αυτά τα μοντέλα διαρκώς ανανεώνονται, ώστε συχνά ξεπερνούν τις δυνατότητες των υπαρχόντων προσομοιωτών. Ταυτόχρονα ο υπολογιστικός φόρτος των μεγάλων δικτύων επιβάλλει τη χρήση τεχνολογίας υψηλής επίδοσης, η οποία εισάγει πολύπλοκα τεχνικά ζητήματα.

Η ταχύτητα της *in silico* ερευνητικής διαδικασίας εξαρτάται από τα επαναλαμβανόμενα συστατικά στάδια κατασκευής *in silico* πειράματος, προσομοίωσής του και εξαγωγής αποτελεσμάτων. Το πρώτο στάδιο μπορεί να επιταχυνθεί με αποδοτικά εργαλεία σχεδίασης μοντέλου και πειράματος, το δεύτερο στάδιο με βελτιώσεις στο υλικό και λογισμικό προσομοίωσης και το τρίτο με αυτοματοποίηση της συλλογής αποτελεσμάτων, ενδιάμεσης ανάλυσης και οπτικοποίησής τους με τον τρόπο που οι ερευνητές θέλουν να εξετάσουν κάθε φορά. Σε όλα τα στάδια είναι σημαντικό να υπάρχει ευχρηστία των υπολογιστικών συστημάτων.

Η νευροεπιστημονική έρευνα αναζητεί τρόπους να μοντελοποιηθούν φυσιολογικά φαινόμενα, που δεν εξηγούνται από τα υπάρχοντα μοντέλα. Οι ερευνητές υποθέτουν νέα μοντέλα που ενδεχομένως περιγράφουν αυτά τα φαινόμενα και τα επαληθεύουν αντιπαραβάλλοντας την προσομοίωσή τους με τις πραγματικές μετρήσεις. Όμως, τα περισσότερα υπάρχοντα λογισμικά ταχείας ανάπτυξης μοντέλου επιτρέπουν τις αλλαγές στα εξεταζόμενα μοντέλα, αλλά είναι σχεδιασμένα για μικρά δίκτυα και προσωπικούς υπολογιστές, ή χρειάζονται απαγορευτικό κόπο και τεχνικές γνώσεις για άμεση χρήση από νευροεπιστήμονες. Οι υπάρχοντες χρηστικοί προσομοιωτές μεγάλης κλίμακας, από την άλλη, συχνά αναπαράγουν ένα δεδομένο μοντέλο νευρώνα

ή πλέγμα δικτύωσης, οπότε περιορίζουν αντίστοιχα το φάσμα των νευρικών δικτύων που μπορούν να προσομοιωθούν.

Καθώς αυξάνεται η πολυπλοκότητα των νέων μοντέλων, αυξάνεται παράλληλα και ο υπολογιστικός φόρτος που απαιτούν. Καθώς η διαδικασία έρευνας απαιτεί διαδραστικότητα, τα αποτελέσματα πρέπει να είναι διαθέσιμα σε εύλογο χρονικό διάστημα. Επομένως, η χρήση πρόσφατων μοντέλων οδηγεί σε ανάγκη για περισσότερη υπολογιστική ισχύ. Λόγω τεχνικών περιορισμών, όμως, για να εκμεταλλευθεί η πρόσθετη υπολογιστική ισχύς, πρέπει το λογισμικό προσομοίωσης να έχει σχεδιαστεί κατάλληλα.

Την τελευταία εικοσαετία, ο ρυθμός εκτέλεσης ακολουθιακών προγραμμάτων αυξάνεται με σημαντικά μειωμένο ρυθμό. Ταυτόχρονα, η πυκνότητα υπολογιστικών μηχανισμών ανά ολοκληρωμένο κύκλωμα συνεχίζει να αυξάνεται με αμείωτο ρυθμό, με αποτέλεσμα να βγουν στο προσκήνιο συσκευές παράλληλης επεξεργασίας, σε υλοποιήσεις πολυπύρηνων επεξεργαστών γενικής χρήσης με διανυσματικές λειτουργίες, διανυσματικών επεξεργαστών, και επανασυνδέσιμης λογικής, σε όλο το φάσμα της υπολογιστικής υψηλών επιδόσεων. Επομένως για να εκμεταλλευθεί όλη η διαθέσιμη υπολογιστική ισχύς, οι αλγόριθμοι πρέπει να αναλυθούν ως προς τη ροή δεδομένων ώστε να εκτελούνται ταυτόχρονα όλοι οι ανεξάρτητοι υπολογισμοί.

Μέχρι πρότινος, για τους περισσότερους χρήστες, η πολυπλοκότητα των προσομοιώσεων περιοριζόταν από την υπάρχουσα επένδυσή τους σε υπολογιστικό υλικό. Πρόσφατα, υπολογιστικά κέντρα και επιχειρήσεις άρχισαν να προσφέρουν υπολογιστικούς πόρους ως υπηρεσία-προϊόν. Οι επεξεργαστές, η μνήμη, η σύνδεση δικτύου και λοιποί πόροι ενός υπολογιστικού κόμβου διαμερίζονται σε εικονικά μηχανήματα που χρησιμοποιούνται από τον τελικό χρήστη μέσω σύνδεσης δικτύου.

Καθώς οι πάροχοι υπολογιστικών υπηρεσιών μπορούν να εστιάσουν στην παροχή υπολογιστικής ισχύος, μπορούν να κατέχουν συστήματα μεγαλύτερης ικανότητας απ΄ότι οι χρήστες, και να τα αξιοποιούν πιο αποδοτικά, προσφέροντας μεγαλύτερης κλίμακας και χαμηλότερου κόστους υπολογιστικές δυνατότητες. Η πρόοδος στη σχετική υποδομή οδήγησε στην τυποποίηση των προσφερόμενων εικονικών μηχανημάτων, που επιτρέπει οι εφαρμογές να μεταφέρονται άμεσα σε οποιαδήποτε σχετική υποδομή.

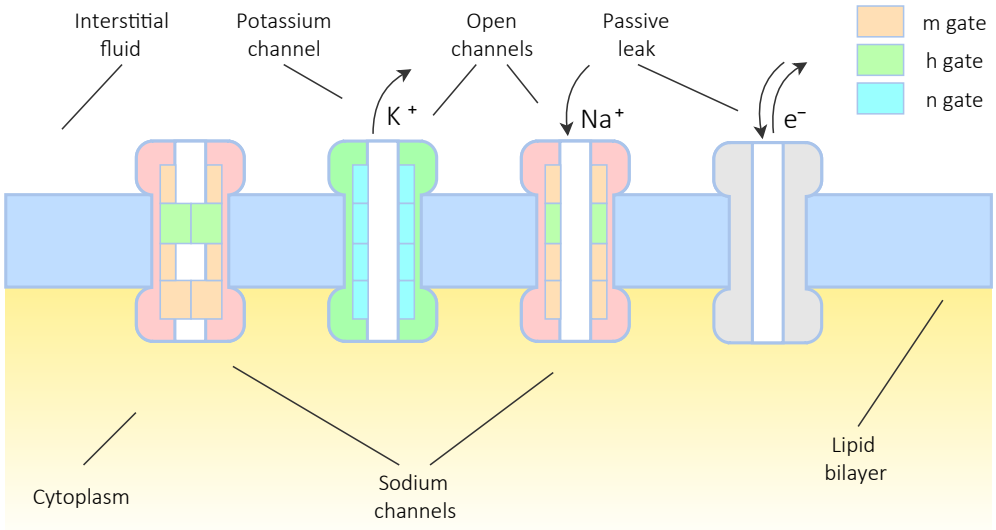
Λόγω της έκτασης της κοινότητας νευροεπιστημόνων που χρησιμοποιούν τα υπάρχοντα λογισμικά προσομοίωσης και της αποδεδειγμένης τους αποτελεσματικότητας, ένα νέο λογισμικό προσομοίωσης πρέπει να μπορεί να συνδυαστεί με αυτά τα εργαλεία και με τις προϋπάρχουσες πρακτικές έρευνας. Καθώς τα πεδία έρευνας είναι αχανή, καινοτόμα και διασυνδεδεμένα, πρέπει να εφαρμοστούν ανεξάρτητα μεταξύ τους εργαλεία, που να ανταλλάσουν αυτόματα πληροφορίες (όπως πειραματικά μοντέλα και αποτελέσματα) μεταξύ τους. Χρειάζεται τα αποτελέσματα των προσομοιώσεων και των πειραμάτων να έχουν ενιαία, κοινά υποστηριζόμενη μορφή ώστε να γίνονται άμεσες συγκρίσεις και επαναχρησιμοποίηση των δεδομένων από άλλα ιδρύματα προς μεγαλύτερο ερευνητικό όφελος. Ακόμη, πρέπει να υπάρχει ενιαία μορφή περιγραφής των μοντέλων, ώστε να δοκιμάζονται τροποποιήσεις που υποστηρίζονται από συγκεκριμένους προσομοιωτές και να μπορούν να χρησιμοποιηθούν τα υπάρχοντα μοντέλα από όλους τους ερευνητές χωρίς κόπο και κίνδυνο λαθών στη μεταφορά.

# Υλοποίηση λύσης

Με βάση τις παραπάνω προδιαγραφές, αναπτύχθηκε ένας προσομοιωτής που δέχεται μοντέλα σε γλώσσα φιλική προς το χρήστη και εφαρμόζει παραλληλισμό για να επιτύχει υψηλές επιδόσεις. Με πρόσθετες επεκτάσεις, μπορεί να τρέχει αυτόνομα ως μια υπηρεσία νέφους, και να ενσωματωθεί στην πλατφόρμα BrainFrame.

Ο προσομοιωτής υλοποιεί τη διαμερισματική μοντελοποίηση Hodgkin-Huxley. Έτσι, ο νευρικός ιστός αποτελείται από νευρώνες που αποτελούνται από διαμερίσματα, ομοιόμορφα ως προς τη χημική κατάσταση. Κάθε διαμέρισμα περιέχει ενδοκυττάριο υγρό, και μεμβράνη που διαχωρίζει το κύτταρο από το περιβάλλον, και περιέχει συμπλέγματα πρωτεϊνών που επιτρέπουν την επιλεκτική μεταφορά ηλεκτρονίων, ιόντων και λοιπών ουσιών από και προς το εσωτερικό του κυττάρου. Συλλογικά τα όμοια συμπλέγματα ονομάζονται κανάλια. Οι πρωτεΐνες που τα αποτελούν μπορούν να είναι ενεργές ή ανενεργές βάσει στοχαστικών διαδικασιών, ενεργοποιώντας αντίστοιχα τα κανάλια στα οποία ανήκουν, και συλλογικά ονομάζονται πύλες ενεργοποίησης των καναλιών ιόντων. Τελικά η δυναμική της ηλεκτροχημικής κατάστασης καθορίζεται από σύστημα συνήθων διαφορικών εξισώσεων. Κάθε νευρώνας στο μοντέλο μπορεί να έχει εντελώς διαφορετική δομή και παραμέτρους.

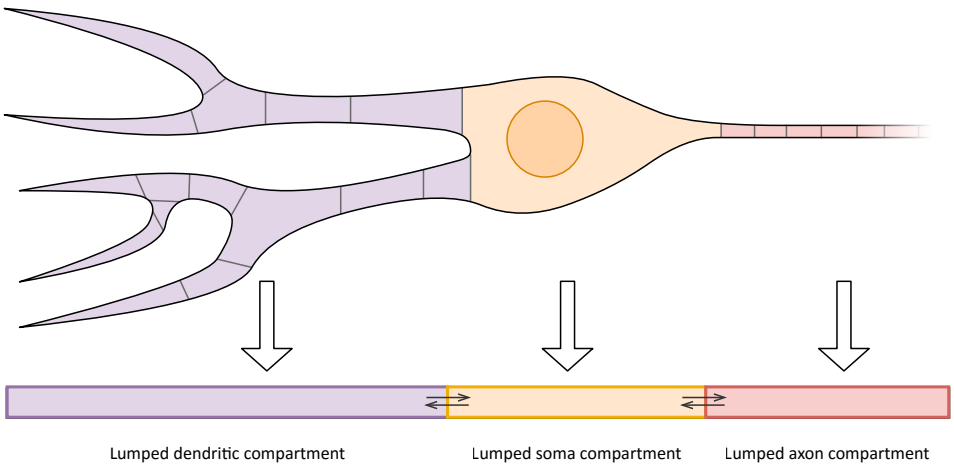
Οι νευρώνες συνδέονται μεταξύ τους με αμφίδρομες ηλεκτρικές συνάψεις, με σταθερή πιθανότητα κάθε πιθανή σύναψη να υπάρχει (δεδομένη πυκνότητα δικτύου). Η εσωτερική αναπαράσταση μπορεί να καλύψει κάθε μελλοντικό μοντέλο.



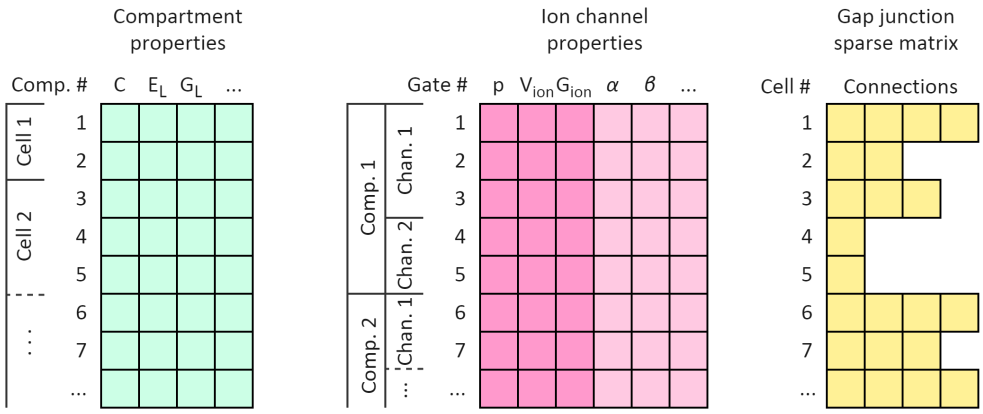

Σχήμα 1: Ενεργά και παθητικά κανάλια στην κυτταρική μεμβράνη, και η επίδραση των πυλών ενεργοποίησης.

Ο προσομοιωτής υποστηρίζει διέγερση των νευρώνων με τετραγωνικό παλμό ρεύματος στους δενδρίτες, του οποίου η ένταση μπορεί να μεταβληθεί ως περιοδική συνάρτηση του αύξοντος αριθμού του κάθε κυττάρου.

Αφού φορτωθεί το μοντέλο από τον προσομοιωτή, η συμβολική μορφή μετατρέπεται σε δομές δεδομένων πιο κατάλληλες για τον πυρήνα προσομοίωσης. Οι παράμετροι του κάθε




Σχήμα 2: Το απλοποιημένο αλυσιδωτό μοντέλο διαμερισμάτων που υποστηρίζεται από τον προσομοιωτή.




Σχήμα 3: Οι πίνακες παραμέτρων που χρησιμοποιούνται από τον πυρήνα προσσομοίωσης.

διαμερίσματος περιγράφονται σε δομή πίνακα, με κάθε σειρά να έχει τις παραμέτρους του αντίστοιχου διαμερίσματος. Παρομοίως, ο πίνακας παραμέτρων καναλιών ιόντων αναλύεται σε συνεχόμενες σειρές ανά πύλη, με τις πληροφορίες του ίδιου του καναλιού να αποθηκεύονται στη σειρά της τελευταίας του πύλης. Η δομή των διαμερισμάτων και των πυλών αποκαθίσταται με απλούς πίνακες αθροιστικού πληθυσμού. Οι μεταβλητές κατάστασης των διαμερισμάτων και των καναλιών ιόντων αποθηκεύονται σε διανύσματα-στήλες παρόμοιας δομής με τους πίνακες παραμέτρων. Οι παράμετροι των συνάψεων αποθηκεύονται σε έναν αραιό πίνακα που κρατάει για κάθε νευρώνα, τις συνάψεις που τον συνδέουν με άλλους και για κάθε σύναψη, αριθμό προσυναπτικού νευρώνα και συναπτικό βάρος.

Στη συνέχεια, ο αλγόριθμος υπολογισμού των νέων τιμών κατάστασης των νευρώνων ανά τα επιλεγμένα χρονικά βήματα, μέσω ολοκλήρωσης των διαφορικών εξισώσεων που περιγράφηκαν παραπάνω από τον αλγόριθμο Euler, εκτελείται επαναληπτικά μέχρι να ολοκληρωθεί το χρονικό διάστημα της προσομοίωσης. Οι τιμές κατάστασης ανά βήματα εκκαθαρίζονται περιοδικά από ένα προσωρινό χώρο μνήμης, ώστε να σχηματιστεί το πλήρες αρχείο εξόδου της

προσομοίωσης.

Ο προσομοιωτής αναπτύχθηκε για επεξεργαστές γενικής χρήσης. Αυτή η αρχιτεκτονική είναι η καταλληλότερη διότι υποστηρίζεται από όλους τους παρόχους υπηρεσιών νέφους, και μπορεί να χειριστεί οποιονδήποτε τύπο μοντέλου άμεσα και αποδοτικά, σε αντίθεση με εξειδικευμένο υλικό.

Στον αλγόριθμο βήματος της προσομοίωσης που περιγράφηκε παραπάνω, προστέθηκαν οδηγίες OpenMP, ώστε να εκμεταλλευθεί ο παραλληλισμός των πολυεπεξεργαστικών συστημάτων. Ο παραλληλισμός προϋποθέτει οτι δεν υπάρχει άμεση εξάρτηση μεταξύ των παράλληλων μερών, και οτι τα παράλληλα μέρη είναι αρκετά μικρά για να μοιραστεί ο φόρτος ομοιόμορφα, και αρκετά μεγάλα ώστε να μην είναι σημαντική η καθυστέρηση συγχρονισμού.

Καθώς τα ρεαλιστικά μοντέλα συνδεσιμότητας νευρώνων έχουν εξαιρετικά μικρή διάμετρο, δεν έχει νόημα η χαλάρωση του παραλληλισμού ως προς το χρόνο. Ακόμη η μη γραμμική, χαοτική δυναμική των μοντέλων νευρώνων αποκλείει την εφαρμογή του αλγορίθμου Parareal[Bed+16]. Επομένως επιλέχθηκε σειριακή εκτέλεση των επιμέρους βημάτων χρόνου.

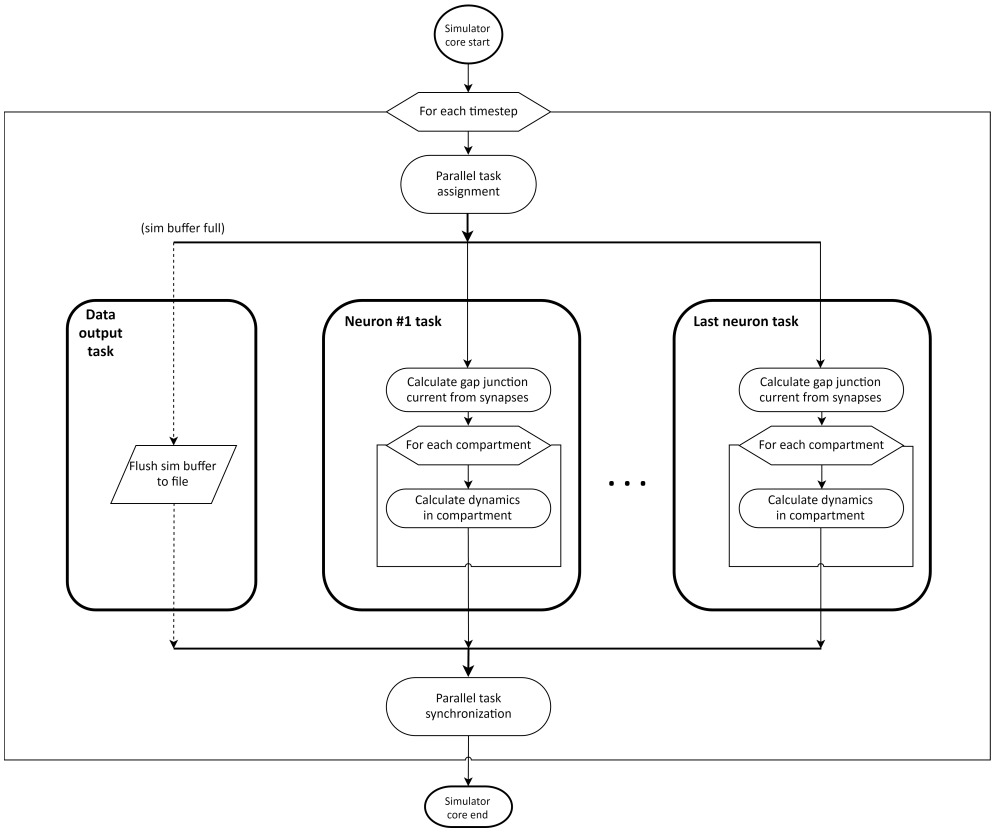
Καθώς σε κάθε βήμα χρόνου, οι επόμενες τιμές που προβλέπει ο ολοκληρωτής Euler μπορούν να παραλληλιστούν ανεξάρτητα, ο παραλληλισμός μπορεί να εφαρμοστεί στον υπολογισμό της καθεμίας χωριστά. Επιλέχθηκε όμως παραλληλισμός σε επίπεδο νευρώνα, αφού έτσι οι υπο-εργασίες είναι μεγάλες αλλά ακόμη ομοιόμορφες, και η επικοινωνία μεταξύ των εργασιών εξαρτάται μόνο από τις συνάψεις των νευρώνων. Μία ακόμη εργασία είναι η εκκαθάριση της μνήμης προσφάτως υπολογισμένων βημάτων, και δρομολογείται παράλληλα με τους υπολογισμούς των νευρώνων.

Ο προσομοιωτής, πέρα από τη χρήση του ως αυτόνομο πρόγραμμα προσωπικού υπολογιστή, μπορεί εύκολα να τρέξει σε συστήματα υψηλής επίδοσης, υπολογιστικού νέφους και σε υπάρχουσες στοίβες λογισμικού έρευνας σε μορφή ζονταινερ. Αυτή η μορφή απομονώνει το απαιτούμενο από το λογισμικό περιβάλλον, ώστε να μην επηρεάζει την υπάρχουσα εγκατάσταση. Το περιβάλλον του ζονταινερ περιγράφεται από ένα αρχικό στιγμιότυπο δεδομένων, που αναπαράγεται ίδιο σε κάθε ζονταινερ που εκκινείται σε κάθε εγκατάσταση. Επιλέχθηκε η πλατφόρμα Δοσκερ, που αναλαμβάνει αυτόματα την απομονωμένη εκτέλεση, εκκίνηση, διαγραφή, επανεκκίνηση, κλωνοποίηση και λοιπές διαχειριστικές λειτουργίες των ζονταινερς.

Για μετατροπή του προσομοιωτή σε αυτόνομη υπηρεσία που συνδέεται χαλαρά με άλλα προγράμματα, προστέθηκε ένας μηχανισμός απομονωμένης εκτέλεσης προγραμμάτων και εξαγωγής αρχείων εξόδου, που αναπτύχθηκε από το εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων. Αυτός ο μηχανισμός αναλαμβάνει να μεταφέρει τα αρχεία εισόδου στο περιβάλλον του προσομοιωτή, να τρέξει το προσομοιωτή και να στείλει τα αρχεία εξόδου στη διεργασία-πελάτη, με χρήση WebSockets ώστε αρκεί μόνο μία TCP σύνδεση για χρήση του ζονταινερ. Έτσι σχηματίζεται μια υπηρεσία ουδέτερη σε περιβάλλον εγκατάστασης και εύκολη και ευέλικτη στη χρήση από εξωτερικό λογισμικό.

Το περιβάλλον του ζονταινερ προσομοιωτή περιέχει το εκτελέσιμο αρχείο καθώς και τις βιβλιοθήκες που χρησιμοποιεί ο προσομοιωτής, καθώς και μια ελαφρώς παραμετροποιημένη έκδοση της υπηρεσίας.

Η υπό ανάπτυξη πλατφόρμα BrainFrame έχει στόχο να παρέχει στους νευροεπιστήμονες



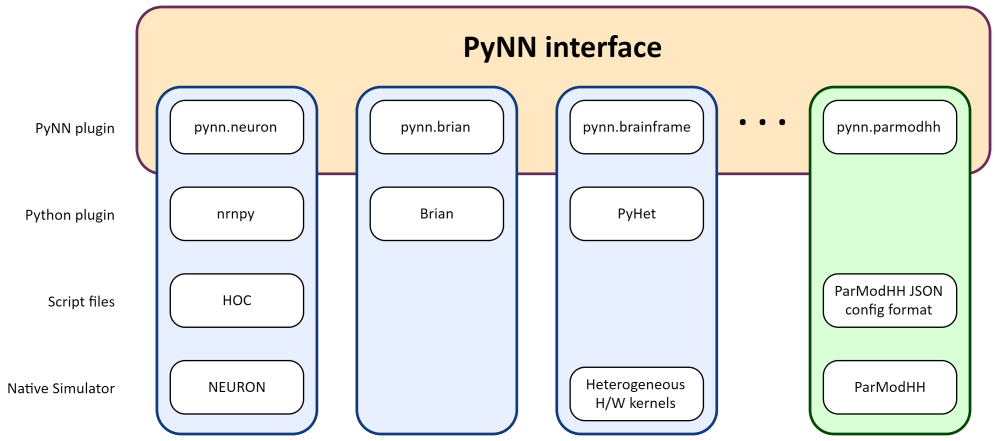

Σχήμα 4: Το διάγραμμα ροής του παράλληλου πυρήνα προσομοίωσης.

δυνατότητα προσομοίωσης πρωτότυπων μοντέλων σε υποδομές υψηλής επίδοσης, ώστε να μπορούν να παράγουν έρευνα πρώτης γραμμής αποδοτικά και χωρίς να τους απασχολούν τα τεχνικά ζητήματα του εκάστοτε υπολογιστικού συστήματος. Ο χρήστης αρκεί να καθορίσει το μοντέλο και τις παραμέτρους της προσομοίωσης, και η πλατφόρμα εκτελεί αυτόματα την προσομοίωση στη βέλτιστη αρχιτεκτονική υλικού και παραδίδει τα αποτελέσματα στο χρήστη. Διαφορετικοί προσομοιωτές υποστηρίζονται στο σύστημα μέσω της βιβλιοθήκης PyNN που παρέχει την κοινή διεπαφή σε αυτούς.

Μία νέα δυνατότητα που προστίθεται στην πλατφόρμα BrainFrame είναι η υλοποίηση των επιμέρους λειτουργικών μερών της πλατφόρμας, ως πράκτορες που υλοποιούν τα αντίστοιχα λειτουργικά μέρη και έχουν δικούς τους υπολογιστικούς πόρους. Παραπάνω από ένας πράκτορας μπορεί να υλοποιεί ένα δεδομένο λειτουργικό μέρος, οπότε ο υπολογιστικός φόρτος για κάθε λειτουργία μπορεί να μοιραστεί μεταξύ των αντίστοιχων πρακτόρων. Με αυτό τον τρόπο, η πλατφόρμα να μπορεί να εκμεταλλευτεί τις υπάρχουσες υποδομές υπολογιστικού νέφους , και να δεσμεύει τους πόρους που χρειάζονται κάθε στιγμή για να καλυφθούν οι ανάγκες των χρηστών.

Ο προσομοιωτής της εργασίας μπορεί να ενσωματωθεί την πλατφόρμα BrainFrame μέσω του PyNN, προσφέροντας ευέλικτη προσομοίωση μοντέλων στο επίπεδο πυλών καναλιών ιόντων. Για τη διεπαφή επεκτείνονται υπάρχουσες κλάσεις αντικειμένων του PyNN, και προστίθενται εντελώς νέες που εκφράζουν τις λεπτομέρειες του μοντέλου που μόνο αυτός κα-

λύπτει. Για να εκτελεστεί η προσομοίωση, το μοντέλο που περιγράφεται στο PyNN τελικά μεταφράζεται στη μορφή JSON που υποστηρίζει ο προσομοιωτής σε προσωρινό αρχείο και μετά εκτελείται το πρόγραμμα προσομοιωτή με επιλογή του παραπάνω αρχείου από τη γραμμή εντολών.




Σχήμα 5: Η προτεινόμενη επέκταση του προσομοιωτή, σε παράθεση με τις υπάρχουσες επεκτάσεις του PyNN.

# Αποτελέσματα

Για να εξεταστεί η κλιμακωσιμότητα του προσομοιωτή, εκτελέστηκε μια σειρά δοκιμών επίδοσης του προσομοιωτή στην υποδομή Amazon Elastic Cloud. Χρησιμοποιήθηκε η σειρά `c5` εικονικών μηχανών του Amazon Elastic Cloud. Αυτή η σειρά έχει δεδομένη μικροαρχιτεκτονική επεξεργαστή που εκτελεί το λογισμικό των εικονικών μηχανών, στο φυσικό επίπεδο. Τα «μεγέθη» εικονικών μηχανών που χρησιμοποιήθηκαν ήταν `c5.xlarge`, `c5.2xlarge`, `c5.4xlarge`, `c5.9xlarge`, `c5.18xlarge`, που έχουν 4, 8, 16, 36, 72 διαθέσιμες vCPUs αντίστοιχα. Το μοντέλο επεξεργαστή που χρησιμοποιήθηκε σε κάθε περίπτωση ήταν Intel Xeon Platinum 8124M, με ονομαστική συχνότητα λειτουργίας 3.0 GHz. Πρέπει να σημειωθεί οτι χρησιμοποιείται η τεχνολογία HyperThreading, οπότε κάθε vCPU αντιστοιχεί σε ένα υπονήμα κάθε πυρήνα.

Σε κάθε δοκιμαστική εκτέλεση του προσομοιωτή, ο δείκτης επίδοσης ήταν ο χρόνος για εκτέλεση 10.000 χρονικών βημάτων της προσομοίωσης.Η έξοδος δεδομένων προσομοίωσης ήταν απενεργοποιημένη, και η εσωτερική αναπαράσταση του πίνακα συνδεσιμότητας ορίστηκε σε αραιή, για άμεση συγκρισιμότητα της αλγοριθμικής επίδοσης για κάθε συνδυασμό παραμέτρων εκτέλεσης. Για εξέταση της επίδοσης του προσομοιωτή ως προς την πολυπλοκότητα του κάθε νευρώνα, κατασκευάστηκαν συνθετικά μοντέλα 1, 2, 4, 8 και 16 διαμερισμάτων ανά νευρώνα και 1, 2, 4 και 8 πυλών καναλιών ιόντων ανά διαμέρισμα.

Μια προσομοίωση έτρεξε για κάθε συνδυασμό παραμέτρων από τα εξής εύρη: 4, 8, 16, 36, 72 διαθέσιμα νήματα, 1.000, 2.000, 4.000, 8.000, 16.000 νευρώνες στο δίκτυο, πυκνότητα συνάψεων 0%, 25%, 50%, 100% όλων των πιθανών, 1, 2, 4, 8, 16 διαμερίσματα ανά νευρώνα

και 1, 2, 4, 8 πύλες καναλιών ιόντων ανά διαμέρισμα. Συνολικά έτρεξαν 2.000 δοκιμαστικά πειράματα για να καλύψουν τους συνδυασμούς παραμέτρων. Τα αποτελέσματα παρουσιάζονται στους παρακάτω πίνακες και σχήματα, για μέγιστη εσωτερική πολυπλοκότητα νευρώνα (καθώς αυτή είναι η κατεύθυνση που κινείται η μελέτη βιολογικών νευρικών δικτύων). Επίσης παρουσιάζεται ενδεικτικά η επίδοση ως συνάρτηση του αριθμού νημάτων για μέγιστο πληθυσμό, και ως συνάρτηση του αριθμού διαμερισμάτων νευρώνων για 8.000 νευρώνες και 25% πυκνότητα.

Πίνακας 1: Χρόνος εκτέλεσης για 4 νήματα, σε δευτερόλεπτα.

| Πυκνότητα δικτύου | Πληθυσμός νευρώνων | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 24.0 | 47.8 | 95.5 | 190.5 | 380.6 |
| 25% | 30.5 | 74.0 | 198.0 | 684.5 | 3296.6 |
| 50% | 36.4 | 96.6 | 292.8 | 994.1 | 5292.3 |
| 100% | 46.5 | 137.9 | 458.2 | 1567.3 | 8822.3 |

Πίνακας 2: Χρόνος εκτέλεσης για 8 νήματα, σε δευτερόλεπτα.

| Πυκνότητα δικτύου | Πληθυσμός νευρώνων | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 11.8 | 23.8 | 47.4 | 95.4 | 191.3 |
| 25% | 14.9 | 35.9 | 95.5 | 349.3 | 1695.8 |
| 50% | 17.7 | 47.8 | 141.2 | 507.0 | 2682.0 |
| 100% | 22.7 | 67.2 | 219.9 | 786.0 | 4432.8 |

Πίνακας 3: Χρόνος εκτέλεσης για 16 νήματα, σε δευτερόλεπτα.

| Πυκνότητα δικτύου | Πληθυσμός νευρώνων | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 6.0 | 12.3 | 24.6 | 48.0 | 95.9 |
| 25% | 7.6 | 18.7 | 49.5 | 173.9 | 831.0 |
| 50% | 9.2 | 24.6 | 72.6 | 246.4 | 1317.8 |
| 100% | 11.8 | 35.3 | 114.2 | 393.0 | 2212.7 |

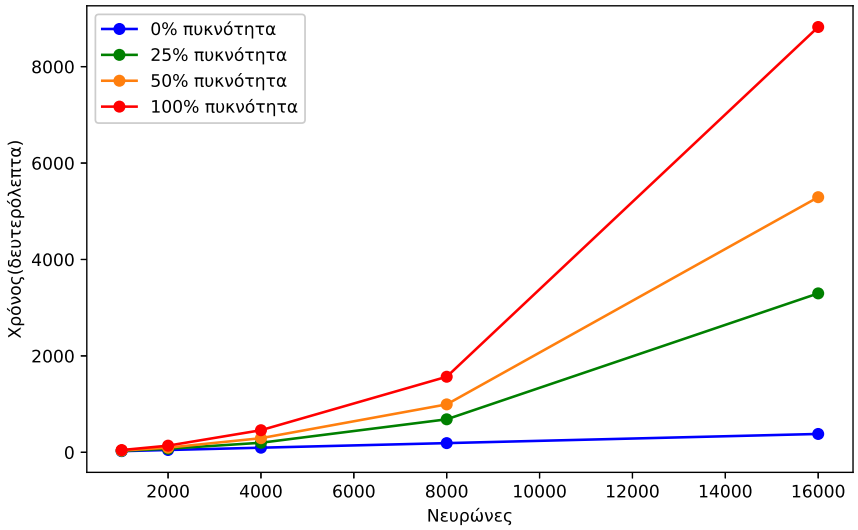
Παρατηρούμε οτι ο προσομοιωτής παρουσιάζει ιδανική ισχυρή κλιμακωσιμότητα σε όλο το εύρος παραμέτρων που εξετάστηκε. Αυτό σημαίνει οτι το υπολογιστικό φορτίο μοιράστηκε εντελώς ομοιόμορφα στα νήματα επεξεργασίας και ο χρόνος μη παράλληλων υπολογισμών, εκκίνησης των νημάτων και συγχρονισμού στο τέλος κάθε επανάληψης ήταν αμελητέος. Παρατηρούμε ακόμη οτι ο χρόνος εκτέλεσης είναι γραμμικός ως προς τον πληθυσμό για απουσία συνάψεων, και τετραγωνικός ως προς το πληθυσμό και γραμμικός ως προς την πυκνότητα

Πίνακας 4: Χρόνος εκτέλεσης για 36 νήματα, σε δευτερόλεπτα.

| Πυκνότητα δικτύου | Πληθυσμός νευρώνων | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 2.7 | 5.4 | 11.2 | 22.2 | 44.2 |
| 25% | 3.4 | 8.5 | 22.3 | 82.8 | 389.7 |
| 50% | 4.1 | 11.4 | 32.4 | 117.1 | 617.5 |
| 100% | 5.3 | 15.9 | 50.6 | 182.4 | 1006.7 |

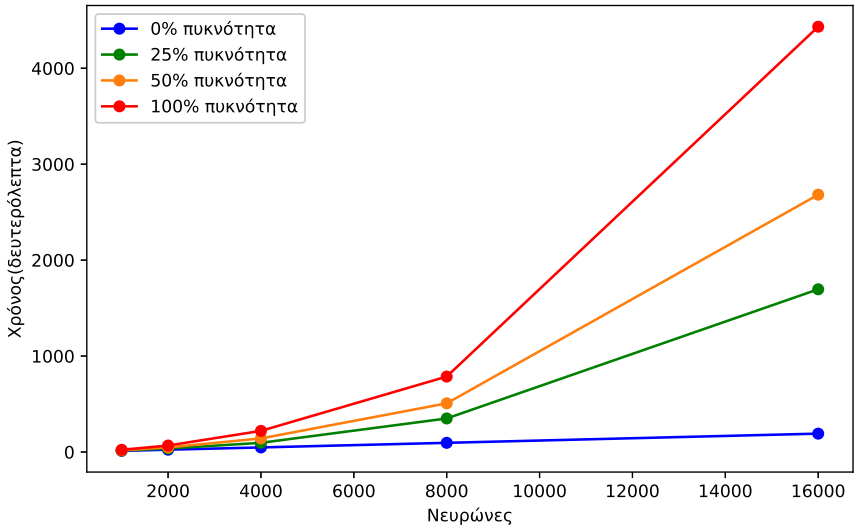
Πίνακας 5: Χρόνος εκτέλεσης για 72 νήματα, σε δευτερόλεπτα.

| Πυκνότητα δικτύου | Πληθυσμός νευρώνων | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 1.5 | 2.9 | 5.7 | 11.9 | 25.9 |
| 25% | 2.4 | 4.4 | 11.9 | 42.4 | 195.8 |
| 50% | 3.1 | 6.2 | 17.6 | 57.8 | 300.7 |
| 100% | 3.2 | 8.1 | 26.4 | 94.1 | 509.4 |

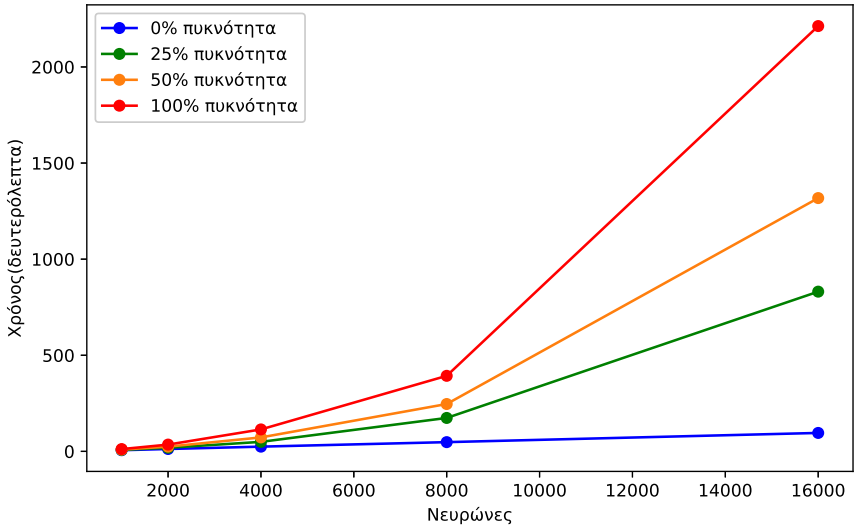



Σχήμα 6: Χρόνος εκτέλεσης για 4 νήματα, συναρτήσει του πληθυσμού νευρώνων.

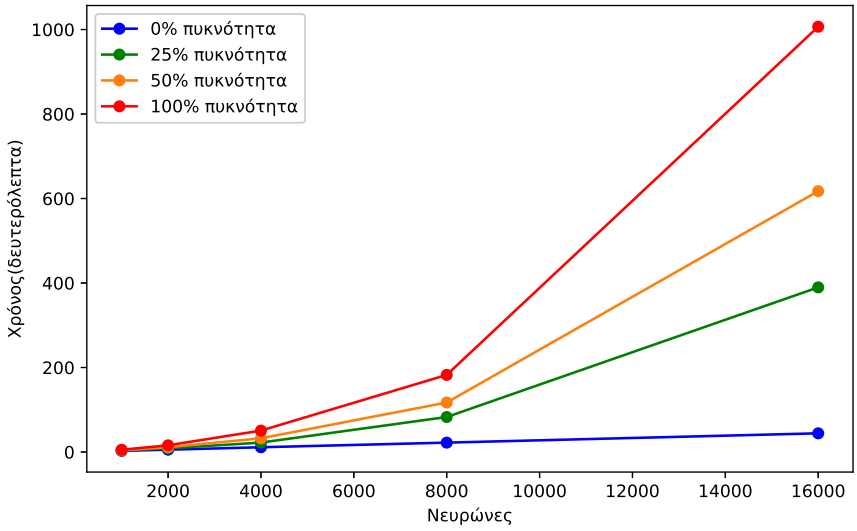
αλλιώς, όπως προβλέπει μια ασυμπτωτική ανάλυση του σειριακού αλγορίθμου. ($O(N * k)$ υπολογισμοί ανά νευρώνα, $O(N^2)$ συνάψεις για μοντέλο σταθερής πιθανότητας συνάψεων)



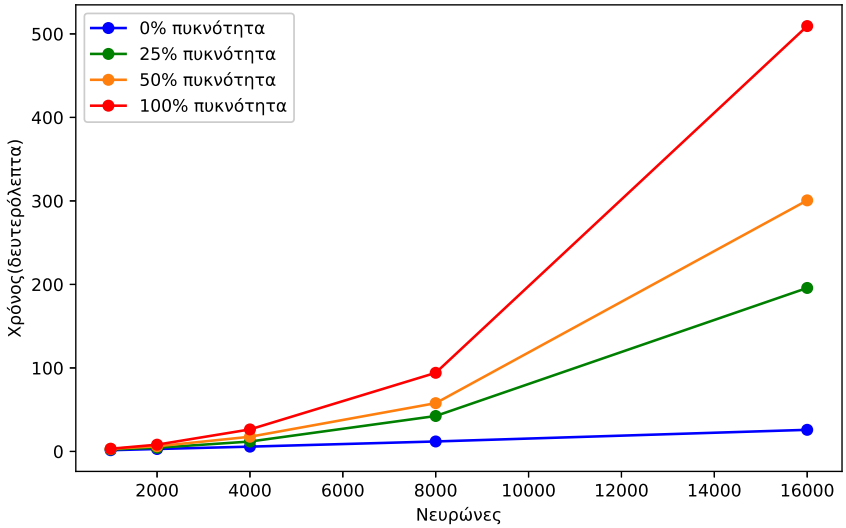

Σχήμα 7: Χρόνος εκτέλεσης για 8 νήματα, συναρτήσει του πληθυσμού νευρώνων.



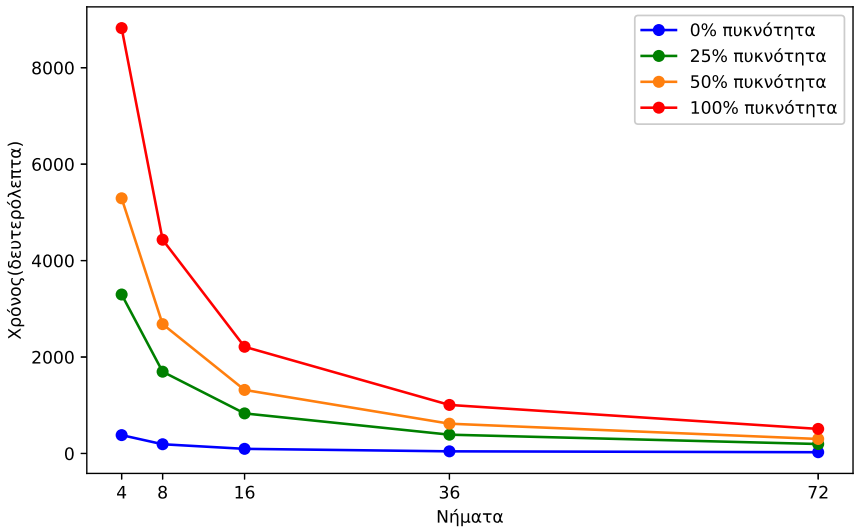

Σχήμα 8: Χρόνος εκτέλεσης για 16 νήματα, συναρτήσει του πληθυσμού νευρώνων.



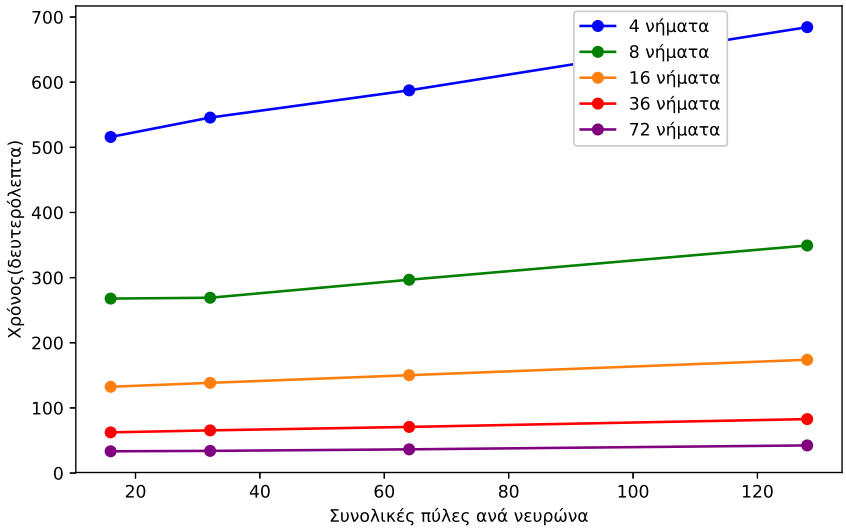

Σχήμα 9: Χρόνος εκτέλεσης για 6 νήματα, συναρτήσει του πληθυσμού νευρώνων.




Σχήμα 10: Χρόνος εκτέλεσης για 72 νήματα, συναρτήσει του πληθυσμού νευρώνων.




Σχήμα 11: Χρόνος εκτέλεσης για δίκτυο 16000 νευρώνων.




Σχήμα 12: Χρόνος εκτέλεσης για δίκτυο 8000 νευρώνων ανδ 25% της μέγιστης πυκνότητας, για 8 πύλες ιόντων ανά διαμέρισμα, ως συνάρτηση του αριθμού διαμερισμάτων.

# Chapter 1

# Introduction

Neuroscience is the scientific field studying the properties of the nervous systems of living beings. Advances in neurology have led to remarkable advances into most fields of medicine, especially cardiology, neurology and psychology, and allowed achievements such as brain mapping, diagnosis through electrocardiography and electroenchephalography, heart and brain pacemakers and deep brain stimulation.

In addition, the study of neural net functions has led to insights in the computer science field, inspiring artificial neural network models and their applications in machine learning and next generation expert systems.

A core tool in modern neuroscience research is computer simulation, providing result estimations for planned experiments[CCT11]. These estimations allow refinement of experimentation plans, by predictively outlining the area of interest in the experimentation space. Using these predictions, researchers can concentrate experimentation on more informative cases, improving value per experiment done[Sil+17] and increasing the efficiency of the research process.

Some neural regions cannot be directly observed, due to practical reasons such as their physical dimensions and the side effects of sedation in *in vivo* studies[AB17]. In this case, computer simulation can still provide qualitative predictions about the behaviour of these systems. Enhancement of the biology research process through computer simulation is called "*in silico* medicine" and has been widely applied in various fields such as pharmacology, radiology, microbiology and regenerative medicine.

Another recent development, in the computer chip technology field, is the end of Dennard scaling[Boh07], which means that sequential algorithms, the traditional way to develop general purpose software, can no longer be accelerated at the explosive rate of the previous decades of improvement in chip technology. In order to fully use the capabilities of newer computers, simulation algorithms must specify all calculations that can be performed independently to the processor, so they can be computed simultaneously whenever possible by the parallel processing machinery available. Due to the non-trivial problem of determining computation dependencies for an arbitrary program, in practice a human expert has to explicitly add parallel programming constructs to a computer program, to effectively utilize available parallel processing features. As a result, as the neural net models under exam-

ination grow larger in complexity, the corresponding computational load of a simulation requires expert work to be fully accelerated on modern computing hardware.

This thesis covers:

- developing a simulator of biological neural nets, at the extended Hodgkin-Huxley level of modelling, with configurable multi-compartment structure and ion channel dynamics, and arbitrary synaptic connectivity
- implementing code parallelism so simulation speed scales with available computational resources
- running performance tests on Amazon Elastic Cloud compute-intensive nodes, for various CPU core counts and types of simulation complexity
- packaging the simulation software into a cloud-ready containerized microservice, and integrating the simulator functionality into an existing neural simulation heterogeneous acceleration platform.

This chapter describes the purpose of this thesis and presents a high-level introduction to the concepts of large-scale computing and simulation on life sciences, and more specifically computational neuroscience. The second chapter lists recent innovative neural simulation technologies, along with their underlying philosophies. The third chapter describes in further detail considerations of combining high-performance and cloud computing to aid physiological research of large, heterogeneous neural nets, which is the the specific use case concerning this thesis. The fourth chapter describes the technical approach and details of the simulation software developed. The fifth chapter presents and interprets performance metrics of the simulator, measured over various degrees of multiprocessing and model complexity factors. Finally, the sixth chapter recapitulates the author's conclusions and suggests further extensions to the present work.

## 1.1 Tools and Techniques in Large-scale Computing

*Large-scale computing* is a term describing applications that conduct one or more computational tasks, using a large-scale computer system. These tasks can belong to a coordinated effort, or they can be completely independent. Modern large-scale computing applications can be loosely grouped in two, not entirely distinct groups, according to the kind of tasks they conduct, or the kind of computations they perform: the *cloud computing* group and the *high-performance computing* group. In the following sections, the differences and similarities between the approaches defining these two groups are described, and the applicability factors of each approach, as well as hybrid attempts attempting to bridge the gap between the two groups, are discussed.

### 1.1.1 Cloud computing

The large-scale production of general-purpose computers, combined with rapid development of Internet infrastructure in the last decades, enabled enterprises centered around high-throughput processing, such as online marketplaces and search engines. These enterprises required a large investment in computer hardware to handle peak load conditions. Since peak load conditions are not frequent, a large part of available capacity stayed unutilized. Efforts there thus made to commercialize surplus computational power, through encapsulation of the underlying hardware into a common model, and development of the necessary usage metering and quality-of-service mechanisms to provide this capacity as a product to end users. As the online computational capacity market matured, common standards were proposed and implemented by vendors globally, with respect to the virtual computational environment exposed to user software, such as virtual machines, networking and clustering facilities[Doc; Kub; Cor; Lin; Opea; Opec]. As a result, a client can now implement software services, hosted entirely on interchangeable, vendor-independent, dynamically-sized infrastructure. This service implementation paradigm is collectively called "cloud computing".[MG+09]

Hosting private data and code in shared underlying resources, and dynamic horizontal scaling of the provided service work load through replication of server node instances are central ideas in cloud computing implementations and applications. Since computational power is treated as a commodity provided by interchangeable physical computers, service-providing computational nodes are allocated ephemerally, so that available capacity can track computational demand for each deployed service. This is a paradigm shift from traditional on-premises systems administration, in search of more flexible computational capacity and operating costs across peaks and trenches of application load.

Under traditional on-premises administration, each application is installed on specific physical machines, which are specially configured to support the applications it runs. Thus, a change in application software or available hardware requires changing the deployment per machine plan, and manually undertaken installation and configuration of new software on the target machines. Although this technique allows for optimal configuration of IT resources, it also demands significant system administration work. As computational load shifts across application, new hardware must be introduced to and removed from the local deployments, inducing continuous deployment and maintenance costs. In contrast, cloud computing vendors outsource the infrastructure-level management costs of all hosted applications, allowing the client's IT personnel to focus on application-level management.

In order to develop a new application for, or migrate an existing application to, cloud computing infrastructure, software engineering effort must be undertaken to match the way various resources (such as networking and storage) are used by the application, with the standardized way these are exposed in cloud environments. Another consideration, of special importance to large-scale applications, is that multiple instances of the application software, running on separate nodes, must be able to cooperate, in order to share the increasing load of inbound requests. This requirement is much more difficult to implement, since it requires

technical know-how in distributed systems, and assessing the corresponding, non-trivial design decisions during software development. However, this necessary technical investment can also be applied to a possible migration as well, retaining the benefits of smooth installation, automated deployment and reduced day-to-day management costs achieved through cloud environment standards.

### 1.1.2 High performance computing

Historically, electronic computers were first applied in generation of large numeric tables, through repetitive arithmetic operations on a massive scale. This mode of computing focuses on the amount of arithmetic operations that can be performed in a given time duration, under predetermined, non-interactive computational tasks, and its manifestations are collectively named "high-performance computing". Top-tier high-performance computing machines are conventionally called supercomputers. The operating features of high-performance computing are extremely common in scientific research, in the form of technical computing; numerically simulating phenomena present in mechanics, electromagnetism, meteorology, ecology and medicine, through approximate models of the specimens under study.

The architecture of high-performance computing systems is quite different from a typical general-purpose computer (though they may incorporate general-purpose computers as submodules). In order to perform arithmetic calculations on a massive scale, they comprise of hundreds or thousands of spatially distinct computational nodes that can operate simultaneously. These computational nodes typically are a combination of special intercommunication hardware, local high-speed memory and one or more general-purpose processors, and often include highly specialized accelerators, greatly enhancing performance of specific commonly used operations, such as signal processing and linear algebra algorithms. A characteristic common, flexible and mass-produced accelerator option present on top-performance supercomputers is GPGPUs.

In practice, when the computational load for simulating a single model is shared among a supercomputer's nodes, these nodes need to communicate with each other to calculate results. The type and scale of the communication load a supercomputer's node interconnection network can sustain is a major factor in the type and size of tasks it can efficiently run, while a high degree of connectivity is limited by technological, size and energy constraints. Therefore, supercomputer implementations arrange computing nodes in an interconnected structure that better fits the applications planned for it. For example, a two-dimensional regular connectivity grid is used for simulation of flat two-dimensional models, such as those encountered in meteorology and geology; since a simulated spot in these models interacts only with physically adjacent regions, a mapping of simulated regions to respective points in the node grid means these nodes only need to communicate with adjacent nodes in the grid, achieving minimal communication delays. For the same reasons, a three-dimensional grid is a natural fit for three-dimensional, dense models. Often, multi-dimensional torus[ASS09], crossbar[Art+15] and fat tree[Lia+14; Ari] network topologies are employed, in a compro-

mise between general task performance and physical limitations.

Due to the important technical decisions and uniqueness of existing implementations, applications targeting a supercomputer must be compatible with the supercomputer's specific architecture, assumptions and limitations, and they also have to be fully optimized against each of its particular capabilities, in order to extract the most computational efficiency possible.

Supercomputer infrastructure is rare, and available run time is scarce due to demand. At the same time, porting a scientific application to a supercomputer's architecture requires significant development effort. As a result, research teams frequently seek to cover their scientific computing needs through alternative solutions , which are more flexible and easy to work with, yet still present high-performance computing characteristics. Typical approaches are using a networked cluster of general-purpose computers, available on-premises or through cloud services vendors, utilizing a specialized accelerator module that best fits the application's processing load, and designing a custom setup combining a workstation or server with multiple, possibly different accelerators, essentially forming a small-scale, ad-hoc high-performance computer. During the last years, cloud services vendors specifically target the high-performance client base with offers that combine physical GPGPUs and reconfigurable logic with the conventional virtual server[Aws; Azua].

### 1.1.3 Tradeoffs and Platform Selection

Present supercomputers enable scientists to simulate various models, at complexity scales and run speeds not possible through other types of solutions. These capabilities are enabled by their extremely powerful node interconnect networks, and the massive amount of specialized accelerator hardware they incorporate. However, in practice their availability is limited; access is typically granted after a formal time-grant procedure concludes, and the combination of occupancy limits and high demand means available time and allocation slots are very limited. As a result, using such systems introduces scheduling delays which directly impact the original goal's progress, regardless of the system's actual run time performance. A second problem with supercomputer practice is the need to fully tailor user software and the underlying computation algorithms to the specific machine in use, increasing non-reusable effort and slowing down the development and research cycles. In conclusion, supercomputers are the best fit for the specific niche of ultimate performance in single complex tasks, when the problem size and complexity exceed what commonly available hardware can practically support.

Cloud computing solutions offer computer facilities as a standard commodity. The application programming interface, such as operating system environment and resource access protocols, are uniform across all vendors. As a result, vendors are largely interchangeable, the pool of available offers greatly widens, and application development, deployment and migration are all made much easier. Important benefits over high-performance computing services are also present. For example, cloud computing can support functionality for an entire interactive system, whereas HPC services offer executions of non-interactive batch

jobs only. Containerized, cloud native applications can be very easily migrated to different vendors and on-premises dedicated setups, whereas migrating an application to a different HPC platform often requires a near-total rewrite to keep system utilization high.

However, traditional cloud services also present significant weaknesses to scientific computing. Analysis of large-scale models, requiring intense communication traffic among computation nodes, are dramatically impacted by mainstream datacenter networking hardware, especially in terms of communication latency which is critical for iterative, synchronous algorithms. Therefore, the high-performance tasks most popular on cloud infrastructure are those that can be distributed to many worker units with minimal cross-communication, such as data analytics, image and video processing, or parameter exploration on large grids. In addition, dynamic worker replication features common in cloud deployments may offer elasticity in allocated computational power, but they require distributed systems knowledge to properly apply.

A direct comparison between the computational power of datacenters and supercomputers needs to account for the different facets of system performance which relevant to each type of application. High interconnect performance requirements, combined with present technology limits, restrict the physical dimensions of typical sized supercomputers. The above, combined with present chip technology limits in computational power per volume and cooling technology limits in thermal power density, put a restriction to total raw computational capacity available. In contrast, datacenter installations can house computer hardware over a much wider system area, since they run diverse, largely unrelated, tasks and communication traffic requirements are subsequently greatly diminished. As a result, while cloud computing infrastructure cannot deliver single-task computation capabilities anywhere close to a supercomputer, it may deliver larger total computational power over a many-task workload, and thereby perform better in distributed computing workloads.

Per the previous arguments, both traditional high-performance computing as well as traditional cloud computing approaches present different advantages and drawbacks in the domain of scientific applications. Therefore, a case can be made for an integrated solution utilizing both cloud and HPC functionality, combining the strengths and minimizing the weaknesses of both paradigms. Already, various cloud vendors offer virtual machines enhanced with acceleration hardware, lowering the level of entry for smaller organizations that would not be able to afford such systems otherwise. At the same time, efforts are being made to port the container paradigm to traditional HPC environments[Ger+17; KSB17]. Since a large part of commonly used scientific software packages is typically available on most supercomputing platforms, another possible approach could be a broker service, receiving computational tasks using those packages from users, and automatically directing them to one out of multiple cloud or HPC services available, in order to minimize expected completion time, system load, or energy consumption. Similar projects have already been launched in the specific field of grid computing, where federation of multiple computational grids has been achieved[Gag04]. A future HPC-oriented solution could also expand the grid computing semantics to include software acceleration in diverse hardware platforms.

## 1.2 Computer Simulation in Life Science - Challenges And Methods

A major, diverse and constantly expanding field of natural science is concerned with the functions of living organisms, from the most fundamental details of metabolism to system studies of the entire bodies of humans and mammals. Aside from the productivity and efficiency benefits simulation can offer, it also allows to investigate many important mechanisms whose properties cannot be isolated in experiments, or even observed directly. Through simulation, the parameters of proposed models can be optimized or implicitly discovered, applying repeated comparisons of available measurement data to model response for each prospective parameter set. As available computational power increases geometrically over time, computer models and techniques are increasingly being introduced into all aspects of life science.

### 1.2.1 Applications

Biomedical applications of computer modelling and simulation revolve around proposing new models, validating proposed models against experimental data, and using verified models to enhance clinical practice.

Tissue modelling has been applied to study the macroscopic mechanical properties of connective tissue, skin and internal organs. Mechanical simulation of large tissue structures has allowed for durability estimation of bones from CT scans, aided the design of aerosol pharmaceutical delivery devices[KZD08], and non-invasive detection of neurodegenerative diseases, as well as computer simulation of surgical operations[Sut+06]. Modelling of less predictable physiological properties, has enabled techniques for prediction of wound and bone fracture healing[Ger14], performance and compatibility of prosthetic and engineered tissue implants[TM16], and sepsis dynamics and prognosis[ANV12]. Computational fluid and soft tissue modelling has produced insights in details of the cardiovascular system[Mor+16].

Molecular dynamics simulations studying the parameters affecting reactivity between pairs of substances were initially developed in general chemistry research. They have been successfully used to model the chemical behaviour of biological structures, such as receptor proteins and ion gates present in the membranes of cells. Such models are crucial to cellular physiological modelling and drug research, and their use allows high-throughput virtual screening of pharmaceutical lead compounds[Jac+07].

In the oncology field, diffusion and absorption models of ionizing radiation allow radiotherapy practitioners to maximize effectiveness and minimize harm[vBa+08]. Available cell models of cancer cells can be macroscopically simulated, providing models that better describe the growth and kinetics of malignant tumours, and aid the refinement of intervention plans[SG17].

Another example of major breakthroughs possible only thorough simulation is the rapid development of advanced artificial pancreas controllers[Man+14]. The introduction of the

FDA-approved UVA/Padova Type 1 diabetes mellitus model has enabled researchers to explore novel blood sugar control laws through insulin injection, evaluating their performance *in silico* instead of running slow, impractical and susceptible to variation tests on non-rodent mammals that were previously required[Pat+09].

### 1.2.2 Requirements

For a simulation system to be of use in research and clinical practice, a set of standards and features must be present. In all cases, accuracy of generated results and compatibility with methods, processes and information systems in use are mandatory, since clinical users have to trust medical-grade equipment, research users have to be sure their hypotheses are not misled by subtle inaccuracies, and in both cases a new simulation system is a new part of an already established process and cannot be used effectively if it is unable to interoperate with the other parts of the process. Usage and simulation speed is also an important factor, since it accelerates the workflow in use and enhanced speed may allow higher simulation quality at the same given run time.

In research use, simulation software must be easily extensible, in order to support new proposed model types and new available hardware, as these two factors are in continuous development, along with the constant progress in research and technology. When the object of study is conception of new models or extension of cutting-edge models, rapid prototyping of model types is critical for research to proceed.

In clinical applications, model modification is not necessary, since medical practice follows already established and proven approaches and protocols. Instead of flexibility, the main productivity factor is velocity in completion of day-to-day routine tasks. A hard requirement for clinical users, that is often unaccounted for in research use, is system and service reliability. A short span of downtime is a minor inconvenience to researchers, but it can have a much more severe impact in a healthcare setting.

### 1.2.3 Approaches

Computer simulations require experimental data on which models can be based, and software that can estimate results for the scenarios under consideration. As a result, the range of available simulations is limited by available data and software packages, and demand for new simulation approaches translates to respective demand for acquisition of new data and development of new software that will make these approaches possible. Globally coordinated projects attempt to standardize production of physiological datasets, and formalize modelling of living organisms, aiming to assemble the entire physiome of each organism[Bas00; Suz+09]. The diversity of medical science has given rise to many different kinds of approaches in simulation, matching the various stages of research progress and clinical adoption.

Under certain models, in both research and clinical use, parameters cannot be estimated from measurement data through a direct analytical algorithm. In these cases, model

parameters are fine-tuned to represent available data, through iterative estimation of model response and parameter optimization to best reproduce the natural response.

In basic research, when entirely new features of physiological activity are being discovered and validated, new models are being developed, either to document new principles being proposed, or attempting to replicate an observed phenomenon. In these cases, the simulator's purpose is to visualize the behaviour under study, and to provide a basis from which more accurate and integrated models will later be developed, as the models are further enriched and validated.

An additional object of basic as well as applied research is modelling and simulation of the various factors degrading measurement data produced by technical systems. For example, metal implants severely impact the results acquired from CT and imagers[Bro+12], and natural brain structures distort magnetic resonance elastography results[McG+].

Another simulation technique is testing new types (or administration protocols) of treatments *in silico* , on current models. This technique examines the results of the simulated experiments, in order to assess their effectiveness and determine which treatments are acceptable for further clinical investigation. A variant of this technique studies the effect of genetic features on the effectiveness of present treatments, combining models describing the interaction of treatments with certain physiological features (such as metabolic pathways[Aro05]), and models correlating genomic data with effects on those features[Jel+15]. Such studies aim to provide drug regimens optimally matching each individual patient. Yet another variant simulates the behaviour of an ensemble of virtual patients under the same virtual experiment, in order to extract macroscopic population models or to get a coarse but wider picture of rare diseases[Car+18].

## 1.3 Computational Neuroscience

Computational neuroscience is the study of the computational functions of neural structures. The focus of this study can be the abstract emergent qualities present in nets of simple functional units, the quantitative physiological function of biological neural tissue, or the high-level features of simpler yet functionally similar approximations of biological models. The various levels of model refinement seek for answers to different questions about functions observed in brains, the most complex, powerful and unexplained neural structures known, and whose capabilities remain outside the reach of present computer technology.

Neurology research is constantly supplying new detailed physiological and anatomic information about the brain. At the time, multiple concurrent research programs all over the world aim to gather physiological data about the human brain's structure and microscopic principles[Amu+16; Jor+15; Jor+15; mPo+16]. For the entire brain and its main regions, the sources from which stimuli arrive and the destinations toward which responses are propagated are documented. New physiological mechanisms affecting neural cell activity are being discovered, and connectomics efforts attempt to provide complete connectivity atlases of entire human brains, down to the synapse level. Using this information, accurate

models can be constructed, representing the brain structures under study. Since living brains are large, compact and fragile structures, *in vivo* measurements can only cover a superficial layer of the whole set of underlying activities taking place.

### 1.3.1 Types of *in silico* experimentation

For computer models to be meaningful, their results must be directly comparable to observed experimental data. Neurological observations can be acquired through experiments *in vivo* , such as EEG scans, measurements regarding involuntary reflexes, and invasive neural potential recording, and experiments in vitro, measuring the electrochemical behaviour of single cells, collections of disassociated cells, and neuronal networks cultivated in dishes[PPM15]. Consequently, efforts to improve and extend existing models, test the effectiveness of experimental plans *in silico* and confidently reuse simulation results as instances of ideal physiological functionality, all need to match simulation output variables with the set of experimentally observable variables, so the models in use can be validated.

Much like with physical experiments, instances of *in silico* experiments are not always identical; certain parameters may be purposely varied between experiments, in order to study how each parameter affects the experiment's outcome. Parameters commonly explored in physical, as well as virtual experiments, are the types of stimuli applied on the neurons(voltage, current, light and their patterns and intensities), the physiological effects of well-known bioactive substances, and structural parameters of the neural net like population size, the density and regularity of synapses, and myelination. In silico experimentation also allows researchers to estimate the results of conditions not possible or practical to control on living specimens, such as rare or critical medical conditions, directly altering the internal chemistry of cells and applying other desirable changes lacking a known way to apply, and the effects of lead compounds on cells.

### 1.3.2 Types of simulation models

The most abstract type of neural modelling is activation models, replacing spiking activity with an aggregate spiking rate. This type of neuron requires the least computational effort to simulate, and is thus used in the largest scales of simulation conducted, and in practical applications of artificial neural networks. These neuron models are often called perceptrons.

A more detailed class of neuron models focuses on the characteristic feature of neuron activity, action potentials(also called spikes in literature), modelling each as a separate discrete event defined by time of occurrence and intensity. The internal mechanism causing spikes is described through deterministic continuous-time, discrete-time or stochastic models, that typically involve a number of internal state variables. These models can describe timing-sensitive behaviours(such as refractory period), causality between activation of different network regions(as observed among functional brain centers), and steady-stare oscillating modes(as those prevalent in deep sleep), while still requiring relatively low computational

resources.

The lowest level of modelling is concerned with the explicit physiological mechanisms at work beneath the neuron's function. These models describe the neuron's state as a set of variables directly mapping to chemical quantities. Each neuron is modelled as one or more physical components comprising the cell, and the continuous-time differential equations prescribing each variable's dynamics are integrated over time. These models expose the most detail and all modelled physiological features to biologists, and are the most accurate way to investigate a natural neuron's behaviour. However, they present a much greater challenge to simulate on present computer hardware, since communication between neurons occurs on each timestep for all neurons, instead of occurring only when action potentials are transmitted. When only the electrical behaviour of cells is required, simplified equivalent voltage models such as the Izhikevich, FitzHugh and Hindmarsch-Rose model can instead be applied with greatly reduced computational effort.

### 1.3.3 Levels of model complexity

Due to their inherent high level of detail, neural structures are being studied in multiple scales of net complexity. Analysis may concern a single neural cell, or even a part of a cell. This level of analysis delves into the way action potentials are propagated through dendritic structures and transmitted across cells, and the ways a cell's internal biochemical mechanisms modulate its electrical state and responses to stimuli over time. Small structures of a few interconnected cells may also be considered, focusing on how a single cell's responses to stimuli are circulated across the cells in the system, how a single neuron's response can be a quite complex function of the other neurons' responses, the formation of stable oscillators and other more complex systemic properties, and how synaptic properties may change over time, as a result of intercommunication patterns[PPM15]. Simulation of massive cell populations causes yet more complex behaviours to manifest and is used to provide insights into the purpose and function of existing brain structures.

# Chapter 2

# Prior Art In Computational Neuroscience

In order to test assumptions about the internal functions of neural tissue, theoretical models have been explored computationally. Typically, each research work uses custom-made simulation tools, targeting the specific problem and questions each group of researchers is currently investigating. Consequently, reusing those tools for further work requires significant technical effort, distracting field researchers from their primary goals. This problem motivated development of general purpose neural simulators, aiming to cover the most common neural features under investigation and most common analysis techniques in use.

Since speed in model and experiment configuration, as well as model simulation, are key to the research cycle, effective research requires collaboration between researchers from both the biomedical and computer engineering disciplines. A recent approach to neuroscientific research is investigation and modelling of large populations of interconnected neurons, since the increased complexity of such nets enables non-trivial emergent behaviours in nature and better describes existing brain structures under study. As the complexity of simulated neural nets increases, computational load to simulate these models increases respectively. Thus, the scale of models that can be explored is limited by existing technology. As a result, novel computer architectures and systems have been devised and applied to enable simulation of more complex, higher quality neural models.

## 2.1 Present neural net simulation methods

The main challenge in neural net simulation is the steep increase in complexity as the cell population increases. Many prior attempts to simulate large nets assign sets of adjacent cells to computational nodes, which then communicate with each other, with interconnection networks inspired from connectivity between cells in various brain regions. This type of neuromorphic hardware essentially avoids the von Neumann barrier[Bac78], forming a distributed computing system. A challenge in simulating existing brain structures stems from the fact that for small regions of brain tissue, about half of the synapses involved originate from cells external to the region, so these models cannot realistically mimic the same

structures *in vivo* [KHO09]. In addition, as the number of distinct computational nodes increases exceeding the amount of synapses of neurons in the model, each neuron tends to require communication with a different node for each of its synapses[Kun+14].

### 2.1.1 Spike message passing implementations

The immense complexity of brain structures has pushed research of high-level functions to use simplified discrete-spike models, in order to get meaningful results for the largest neural nets possible. Implementations have been made on GPU-accelerated desktop computers, on server clusters, on current top-tier supercomputers and on specialized neuromorphic computers, in accordance with the multiple useful levels of network analysis. The all-or-none principle prescribes that all action potentials are functionally equivalent. Thus, a spike's effect to a postsynaptic neuron is determined only by that neuron and properties of the synapse. This means that during simulation, a neuron only needs to be notified about spikes produced by its presynaptic cells, when the spikes occur. As a result, the internal model of each neuron can be improved in fidelity, with total communication traffic remaining practically the same.

#### 2.1.1.1 Custom designs

The massively parallel nature of neural tissue inspired analogous distributed computing designs, approximating their form as well as their function. Since practical performance requirements mandate extremely fast interaction between simulated neurons, these designs had to be implemented on bespoke, low-latency interconnection hardware.

The SpiNNaker project[Kun+14] has developed a mass-scale ASIC-based neuromorphic architecture, comprising of tens of thousands of identical chips, each containing 18 accelerated ARM CPU cores. Chips on the same component board communicate with each other through a novel network operating on asynchronous digital logic, providing quick, broad and energy efficient exchange of spike event messages. Portions of the entire computer are allocated for execution of individual experiments. Recent software improvements have enabled experiments spanning multiple 768-core boards. The ARM cores can be programmed for a wide range of spiking neural network applications, both realistic and artificial.

Another class of ASIC implementations combines analog operation of single neuron models with exchange of discrete spike events, forming a mixed-signal approach. An early endeavour implemented a network of integrate-and-fire neurons, connected with synapses presenting spike-timing-dependent plasticity[Vog+02]. A recent large-scale implementation models analog adaptive exponential integrate and fire neurons, with direct electrical connections between neurons in the same die and spike message passing between dies on the same wafer through digital high-throughput lanes[SFM08].

An alternative approach retaining the degree of customization and dramatically decreasing development cost is using FPGAs. A recent attempt models each cell as a classical Hodgkin-Huxley neuron on a three-dimensional FPGA chip and exploits its on-cell

network for high-volume, low-latency spike traffic, while entirely eliminating multiplier blocks[Yan+18]. Another claims a record amount of neurons simulated in real time, by using the simple leaky integrate-and-fire models, forgoing flexible connectivity models by adopting a cortex-approximating hierarchical structure of neural columns, and completing processing of a different portion of the cell population on each clock cycle, while pipelining the processing stages[WTS18].

#### 2.1.1.2 Conventional computing infrastructure

Another approach to massive-scale spiking neural network simulation utilizes the versatile, mass-produced capabilities of more common hardware used in modern supercomputers as well as consumer computer units. Parallel versions of the highly popular NEURON and NEST neural simulation packages, as well as the C2 simulator, have been used to evaluate pilot models of entire brain structures, on the K computer and Blue Gene series supercomputers[Jor+18; Ana+09]. Algorithms distributing portions of the connectivity matrix among relevant nodes containing the pre-synaptic and post-synaptic neurons achieve linear total memory usage and constant MPI buffer sizes, as the cell population increases with a fixed connectivity degree[Kun+14]. A Beowulf cluster of commodity general purpose computers has been used for simulations based on anatomical data, from slice scans of rat, cat and human brain tissue, containing 22 different types of cells as they are laminated on cortical columns, and spontaneously expressing behavioural regimes of normal brain activity[Ana+09]. In the desktop computer scale, a highly efficient GPU-based simulator is available for Izhikevich neuron models with plastic synapses[Nag+09].

### 2.1.2 Electrophysiological implementations: Compartmental models

Some cases of neuronal network simulation focus on getting results on the underlying physiological features of neural activity, such as *in silico* experiments involving neuromodulators or other neuroactive compounds. These compounds indirectly affect the neuron's chemistry, so their precise chemical effects are an object of deeper investigation. In addition, new electrochemical properties of the cell membrane and its differentiated regions are constantly being discovered[SBS12]. A coarse formulation of these features splits the cell into singular points (also called point neurons) or geometrically separated lumped compartments, whose biophysical state is modelled after ordinary differential equations. A significant challenge at this scale is that the interaction between neurons happens on continuous time, instead of discrete spike events. This means that communication between neurons has much tighter requirements, either in analog or digital simulation.

#### 2.1.2.1 Equivalent electronic circuit implementations

An experimental network chip, implementing analog sodium-potassium cell dynamics and interconnection with programmable synaptic densities was implemented in [FGH06]. A

generic ion channel simulation mechanism using analog MOSFETs was proposed in order to fabricate large populations of realistic cell models in silicon[HB07]. However, the inflexibilities inherent in the medium and the excessive power required for analog interconnection between neurons in neighbouring chips[SFM08] has directed most subsequent large-scale efforts to mixed-signal approaches, communicating through digitized spikes, as mentioned above.

#### 2.1.2.2 Discretized integration implementations

The digital nature of computers lends itself to approximating the ODEs of compartmental neuron models, through sequential progression of the system state over small time deltas. Due to continuous interaction between connected neurons, which is necessary to investigate subthreshold neuronal activity and resulting synchronization features in brain sections[Cho+10], communication of updated state values also has to occur constantly among neurons, whether action potentials are generated or not, greatly increasing intercommunication traffic. Unlike spiking models, an increase in model time resolution directly impacts total data exchanged between neurons. In addition, the differential equations describing biophysically accurate models are much stiffer than those used in spiking models, so they require higher time resolution to simulate, further worsening runtime performance.

The Blue Brain project[Mar06] has used the NEURON package to simulate biophysically accurate models of neocortical neurons on Blue Gene series computers (soon to be superseded by Blue Brain 5, based on Hewlett Packard Enterprise SGI 8600 nodes[Ent]). The impact of increased time resolution was mitigated through a slow rate of voltage updates between neurons, and accelerating resting-state neuron simulation through telescopic projective integration, combining small transient timesteps with longer ones, in a GPU implementation[Wan+11]. The general UC4 simulation package combined with model generation and visualisation software into an integrated simulation platform, allowing investigation of heterogeneous structures of neurons with custom properties and ion channels, with diverse morphologies, at supercomputer scale[Bre+16]. Simulation performance of Inferior Olive model cells with arbitrary gap junction connectivity has been investigated on Xeon Phi manycore processors, clusters of Xeon Phi-accelerated computers, GPUs and reconfigurable dataflow engines[Sma+16]. Finally, the simulator developed for this thesis also follows the compartmental model ODE integration approach.

### 2.1.3 Electrophysiological implementations: Membrane and volume-based models

The biophysical aspect of neuroscience is concerned with the effects occurring across the cell's membrane surface and cytoplasm volume, such as ion diffusion, molecular dynamics and the propagation of action potentials. Such models are described in terms of partial differential equations (or, in the case of molecular dynamics, stochastic processes), which then are approximately solved through finite element analysis, applying discretization in

space and time and solving for the average state of approximate lumps in discrete time steps.

This type of analysis allows for spatially coherent slices of the simulation's volume or surface to be straightforwardly assigned to separate computational nodes on a grid, reducing communication to between nodes of adjacent slices and minimizing its total volume, which makes it a better fit for grid supercomputer architectures. Such models, though, are typically much stiffer on the spatial dimensions, so multiple steps reconciling state values across the model have to be run between time steps. Tissue simulators segmenting neurons in tubular compartments are also included in this section, due to their computational load similarity with finite-element microscopic approaches.

Compartmental volume-based approaches include splitting single neurons into multiple tree-based compartments, implicitly solving the whole neuron's state and reducing spatial integration stiffness[HMS08], and segmenting whole neurons across spatial block borders. The last approach has achieved scaling for up to 10 billion synapses in the Blue Gene/P architecture[KW11]. In small-scale membrane dynamics, a space- and time- parallel approach has been proposed, parallelizing time through the Parareal algorithm[Bed+16]. In the molecular dynamics level, random walk-based ligand diffusion models are parallelized with the spatial grid's nodes exchanging ligands that wander between their assigned zones[Bal+04].

## 2.2 Flexible neural simulation platforms

Another approach, which is particularly useful in simulation of heterogeneous nets, is extremely parametrizable, general purpose neural simulation software. Simulators of this type do not follow the behaviour of a single class of neural net models. Instead, they provide simulation capability for an as diverse as possible range of neural models, from coarse functionality-oriented models to the most detailed molecular biology-oriented models. For each type of model, the aim is to allow users to modify any parameter that could be defined for that type. Under this approach, researchers can run simulations on the most detailed and highest quality neural models available, as well as discover new extensions to the existing types of models, without the need to create a new software or hardware application from scratch. Simulation speed and model complexity can be enhanced using large-scale computing infrastructure, which provides the necessary computing capacity.

### 2.2.1 Wide-range model simulators on personal computers

The demands of neuroscientific research have given rise to a numerous family of neural simulation packages, such as NEURON, NEST, BRIAN, MOOSE, and GENESIS. In addition, general purpose scientific tools such as MATLAB and Mathematica have also been applied to try out novel model types and as a following stage of results processing. Most of these rapid prototyping tools are designed for simple, consumer-grade general-purpose computers and are thus limited to the capabilities of a single desktop computer or workstation. As analysis of large neural nets has been gaining increasing traction over the last

decade, though, popular simulators such as NEURON and NEST have been extended to support large-scale computer systems, although computation is still typically performed on CPUs only, without use of accelerators.

### 2.2.2 Wide-range model simulators on cloud platforms

The limited performance of desktop computers has led researchers to apply high-performance computing platforms on *in silico* experiments, using simulation packages supporting these platforms or custom, model-specific programs. These platforms serve multiple research groups concurrently, assigning a fraction of available hardware to each computational task request. Submitting a task for processing on each platform involves creating an auxiliary automation program that interacts with the platform's queueing mechanism and task execution environment. However, getting access to such systems typically goes through a peer-reviewed grant process, introducing a significant delay to service availability. Also, switching between computing platforms requires many small changes to the "glue" automation, which need effort and technical expertise, reducing productivity and eventually hampering integration of high-performance computing in biomedical research practice. Applying cloud computing principles to existing HPC infrastructure can improve usability and extend the range of available platforms by presenting the computational service in an uniform manner, eliminating friction with platform technicalities.

#### 2.2.2.1 Simulation Platform

The Simulation Platform cloud-based online simulation environment[Yam+11] applies commonly used IaaS[MG+09] and remote desktop technology in use, to provide a desktop environment where commonly used neuroscientific and auxiliary software packages are already preconfigured and available. Many science-oriented Linux distributions have been developed to offer such an environment in lab computer deployments, but Simulation Platform requires just a web browser and modest resources on each client terminal in order to access the desktop environment. This approach leverages the experience users have using existing scientific software, presents the same environment they are familiar with on desktop computers, and enables a fully interactive work process while running on cloud resources. However, model scale and computation performance are limited due to the system's lack of support for cluster parallelism and hardware acceleration.

#### 2.2.2.2 Neuroscience Gateway

The Neuroscience Gateway[Car+14] provides high-performance neural simulation capabilities to end users through a web-based platform. Users can upload simulator-specific experiment definition files to online storage, and then order a simulation run based on these files. The system then undertakes all technical details involved with running the simulation on HPC infrastructure, such as interfacing with each platform's job submission mechanism, optimally configuring simulator software for each platform, and management and transfers

of input and output files. When the simulation run finishes, the user can retrieve the output data from online storage, through the same user interface. A REST API is also available so simulation tasks can be submitted through alternate interfaces.

However, the non-interactive nature of HPC infrastructure means the user has no information feedback as the simulation progresses, and has to wait for the whole simulation run to finish instead. In addition, the interface is not integrated with the rest of the tools researchers use, and even minor edits to the simulation definition require the user to upload the new respective file versions before using them on the platform. Since the platform accepts only simulator-specific files, switching between simulator packages requires creating entirely new simulation definition files. Although the supported PyNN library offers a degree of flexibility in this aspect, the platform's workflow precludes automatic simulation software selection and requires intimate knowledge of the simulation software used.

#### 2.2.2.3 BrainFrame

BrainFrame[Sma+17] is a proposed heterogeneous acceleration platform under development, employing general purpose CPUs as well as various accelerator modules. It aims to provide neuroscientists with a user-friendly model and experiment design interface, through which they can easily run model simulations on the technology that fits each model's computational requirements best, in order to productively match high-performance computing with field research. The radical differences between acceleration hardware platforms enforce the use of different simulation software compatible with each platform. Therefore, a common model definition format, covering the range of *in silico* experiments supported, is required in order to dynamically run the same *in silico* experiments on different technologies.

The BrainFrame platform extends the PyNN framework[BS09] with a frontend plugin that automates conversion between PyNN models and the form each backend simulator requires, and can select the optimal simulator platform for the properties of each given model. Interaction with end users is performed through a user-friendly web interface, providing live information for each simulation run's progress. The technical details of each underlying technology are fully handled by the platform, and backend resources can be dynamically adjusted to user demand using existing cloud infrastructure.

# Chapter 3

# Problem Statement

This thesis examines the application of cloud and high-performance computing in physiological simulation of large heterogeneous neural nets, through an interface usable by field practitioners without special training. In order to be effective, such an application must present a diverse range of features, in the domains of end-user usability, technical capabilities, and meaningful compatibility with processes and software in present use.

## 3.1 Neuroscientific research requirements

In current practice, neuroscientists explore different models of how neural cells operate and interact, in order to better describe the functionality located in various brain areas. These models can be practically explored only through computer simulation, due to their non-linear[HH52] dynamics. Often, these models are being explored for the first time or are not in common use, so they are not readily simulated by existing software.

In addition, large-scale models require a computational load too large to be run on a researcher's personal computer or workstation, and have to run on high-performance infrastructure instead. Running scientific applications on high-performance platforms, though, requires computer engineering expertise to conduct, in terms of necessary modifications to simulator software as well as appropriate system configuration to match the application's load profile. Technical limitations often force end users to make technical decisions outside their field of expertise, distracting them from the task at hand and reducing productivity.

A major factor of researcher productivity is the turnover time between testing hypotheses and obtaining results, which is defined by the required time to simulate hypotheses, time to extract conclusions, and time to plan new *in silico* experiments.

### 3.1.1 Ease of use and research workflow facilitation

As simulation became ubiquitous in neuroscience research, the need arose for a commonly-used language describing anatomic, as well as biophysical properties of neural cells and nets. Such a format amplifies the effectiveness of both simulation and neural imaging efforts, as it provides a common input and output method respectively to the two research communities.

This way, imaging and connectomics projects can offer their results to the same pool of commonly understood physical models, and model researchers can pick any model from the common pool to work on and eventually refine.

A convenient starting point to modelling the cell type under study is to download a similar, already defined neuron model from research databases such as [Hin+04], [Neua]. These models are described in commonly used formats, which can be directly imported into simulation software[Neub].

Frequently, in accordance with widespread demand, high performance computing infrastructure is physically located in individual computing centres, separately from the organizations using the resources. Thus users access the machines through remote access solutions, such as text terminals, graphical windowing environments, client applications offloading computation requests to the infrastructure, and web portals, allowing users to remotely run computation tasks using interactive web pages. The productivity gain from using such solutions is determined by how directly users can perform desired actions (for example, editing model parameters through an interactive form, compared to manually editing multiple text files).

In research, the measure of utility of a tool is the increase in productivity it offers to researchers. A useful toolset in neuroscience research allows users to work productively, by bridging the semantic gap between the physiological aspect of examined models and technical implementation aspect of the underlying tools, and accelerating the workflows in use. The simulation software has to present itself in terms familiar to neurologists, and allow easy completion of common tasks.

The whole *in silico* research process can be accelerated in the experiment input, simulation run and feedback stages; the experiment input stage can be sped up by effective model and experiment design tools, the simulation run stage can be sped up by hardware and software improvements in the simulation engines in use, and the feedback stage can be sped up through automation in compiling the generated results, applying intermediate processing and displaying final results according to the precise way researchers want to examine. Along the entire work cycle, productivity can also be increased through reduced effort and delays in day-to-day usage of computer systems, in each of the above stages.

### 3.1.2 Flexibility in exploring novel cell structures and nets

Neural model researchers explore ways to model *in vivo* or *in vitro* measurement data, that is not yet explained by existing models, through close investigation of the discrepancies between physically measured data and predictions of current models. The aim is to minimize those discrepancies by application of new models. To that end, researchers hypothesise new mechanisms that could explain the unmodelled behaviour, then they put them to the test, initially using *in silico* simulation and subsequently performing more accurate live experiments, and eventually produce new, more realistic models. Literature examples of extending baseline models to better capture observed features of neuron behaviour are [Des+98] and [DG+12].

The process of neuroscience research involves formulating novel neural net models, beyond what is currently codified and standardized by the scientific community.

Present flexible, rapid model development software packages allow for quick exploration of altered models, but they often are not designed to effectively support large scale net simulation. Large net simulation can pose a problem as the underlying algorithms may not scale for large sets of neurons, or the simulators may not support acceleration features of available hardware that could mitigate the increasing computational load.

Existing simulators of large neuron populations, on the other hand, often overcome technical limitations by adopting alternative system architectures, and shaping the computational system in ways mimicking the form of the neural nets to be modelled, in a process named neuromorphic computing[Mea90]. Assumptions made about the simulated neural net's form, though, restrict the range of connectivity and neuron models that can be mapped on the hardware, to specific architectural decisions (such as no support for heterogeneous cell populations in uniform-architecture implementations, restricted or fixed model parameter values in analog implementations). Thus, such simulators offer limited flexibility in modelling of neurons and synapses, and restricts exploration of new models to specific narrow classes, supported by each large-scale simulator.

## 3.2 Scalability of simulation computational load

As the complexity of cutting-edge models increases, so does their computational load. Since research is an interactive process, simulation results need to be available in a practical timeframe. Thus, effective simulation of more complex models requires more available computational power. Due to existing technological limitations, design changes often need to be made to existing simulation software, in order to properly utilize extra computational power.

### 3.2.1 Simulation acceleration through hardware utilization

Since the turn of the century, sequential-instruction program execution speed has been increasing at a greatly diminished rate. Meanwhile, the density of computational machinery density on chips has not stopped increasing. The result is that available computational power is increasingly becoming more than a sequential operation program can utilize. This effect led to a resurgence of parallel-processing devices, through implementations of multiple CPU cores, vector operations, vector processors, and reconfigurable logic devices, to the forefront of general-purpose high-end computing.

For the increase of available computing power to effect an increase in simulation power, therefore, simulation algorithms have to break out of the sequential execution paradigm and exploit the capabilities of increasingly parallel modern computers. The required software changes include a study of interdependence in calculations under each algorithm, so that independent calculations can run simultaneously. In some cases, an optimized serial

algorithm may be replaced by a less theoretically efficient, but more effective in practice, parallel alternative.

### 3.2.2 Dynamic allocation and sharing of computational resources

The complexity of a simulation run is, both in time and scale, limited by computational hardware available to the researchers. Available computational capacity is typically limited by each institution's capital investment in computer hardware.

Recently, supercomputing facilities as well as various commercial enterprises offer surplus computational resources to end users, converted to a computational service product. Through mature, widely-supported virtualization middleware, the processors, random-access memory, non-volatile storage, network transfer throughput and other resources of a physical computer node are securely partitioned into smaller allocation units. A process running on such a partition cannot access or otherwise affect processes running on other allocation units on the same node. As each end user requests a specific type of allocation unit, a free instance is quickly assigned to that user and they get control of its partitioned resources for the requested time duration. Each partition is presented to its assigned end user as a virtual machine, accessed through a secure remote connection.

The end users can, thus, run arbitrary desired software, as long as it is compatible with the virtual machine environment. Users are billed by time using the partition, or by actual resource utilization[Ama; Azub], according to service plan selected. The service provided by such vendors is called " Infrastructure as a Service", as per NIST definition[MG+09], and is one of the service models in the larger domain of cloud computing.

As demand for such services - and specifically for computational power - increased, the range of processors available widened from general-purpose CPUs, to more specialized GPUs and reconfigurable logic, which are often more suited to high-performance computing tasks such as machine learning, large-scale data analytics and image processing, photorealistic graphics rendering and computer-aided design. Also, the focus of IaaS providers on high-end hardware allows them to commission and lease much higher-capacity systems than other organizations could themselves afford. Commodification of computational power and resources can thus lead to more efficient, more expansive and cheaper scientific computing.

Applying Infrastructure as a Service on numeric simulation trades the finer degree of control and possible custom options(such as application-specific hardware) of on-premises infrastructure for a much higher ceiling on peak resource usage(also called hyperscale computing[Pra15]), reduced system maintenance costs, and a possible reduction on total operational costs by paying only for active computation time.

Due to the IaaS market expanding and the corresponding technologies maturing, industry standards have emerged for the virtualized environments available, such as Linux containers[Lin]. Thus, proper porting effort to run simulation software on IaaS platforms can be re-used to run the same software on most other cloud vendors, or an on-premises setup, moving forward.

## 3.3 Interoperability with existing simulation workflows

Due to the popularity of present simulation packages in the neuroscience community and their proven effectiveness, a new simulator must be able to work along the existing software stacks and the established research workflows and processes. A central part of a software package's interoperability is the ability to present its features and functions through terms and concepts already known to research personnel (neural net analysis and experimentation techniques, cell modelling through biological factors, masking unnecessary computer science details), and its ability to exchange information with the rest of the software tools currently in use (such as neural net generators and visualization utilities). These features are necessary for a smooth and effective introduction of a new simulation solution.

### 3.3.1 Incorporating existing software functionality

The neuroscientific research process comprises of many individual tools that work together to provide the final result. Since the field is vast, interdisciplinary and innovative, no integrated solution can cover the entire spectrum of research procedures involved. Therefore, a new tool introduced in a research process must be able to cooperate with the rest of the tools in use. Effective cooperation consists of minimal effort in exchanging data, and enhancement of the whole process through the specific capabilities the tool offers. For example, a model simulation kernel can expand the set of models supported by the simulation frontend, and a model definition processor can allow the end user to describe the desired model in more efficient ways.

### 3.3.2 Simulation data format compatibility

Computer simulators provide the calculated results in the form of data files. Presentation, and comparison between experimental results and model-based estimations, form a major part of the research process. A single simulator cannot always process all models under examination, and simulation results must have the same form that genuine experimental data are available in, to allow for comparative study. Thus, the results files produced by simulators and experimental data acquisition all have to be convertible into a common form that is suitable for further analysis tools. In addition, simulation data files need to be available in a specific uniform, commonly understood format, so they can be immediately used by other research laboratories all over the world and exert a drastically greater impact on scientific progress.

### 3.3.3 Simulation model schema compatibility

An important factor in *in silico* neuroscience is having a common way to represent experiments to be simulated. The development process of new models usually starts from a commonly known model, on which different variations are proposed and examined. When these variations need to be tested on different simulators and no common representation

format is available, the initial model has to be manually converted to the form each simulator recognises, leading to unnecessary, tedious effort and harbours the risk of introducing errors into the model descriptions.

The use of a common representation format also allows researchers to exchange proposed models and effortlessly apply them, accelerating their work and allowing them to survey models under study and development in laboratories worldwide. A standard model, paired with its simulation results, can also be used in order to test a new simulation setup for correctness. This case is very useful for researchers configuring a new simulation setup, or developing a new simulation algorithm or middleware.

Another capability enabled by a common model representation is fully automatic execution of an experiment model on different simulation systems, directly or by converting the representation to each system's native format. This approach has been applied in the BrainFrame platform[Sma+17], so that a specific model of human Inferior Olive neurons can be simulated on different computer architectures. Depending on requested model parameters such as population size and connection density, the platform can automatically select the fastest architecture to simulate the model, achieving significant run time reduction compared to employing a single architecture.

## 3.4 Present work

This thesis proposes a neural net simulator solution on the compartmental extended Hodgkin-Huxley level of analysis, covering the requirements described in this chapter in terms of:

- Flexibility; simulated cells can be defined at a high degree of detail, with regard to cell structure and electro-chemical mechanisms. Each cell can be modelled entirely independently of other cells. Each single cell can be modelled by an arbitrary amount of cell compartments, and an arbitrary amount of ion channels(and respective gates) for each compartment. Synaptic connectivity among the simulated cells can an arbitrary structure.

- Large-scale model support; along with the flexible neural net modelling capabilities described above, the simulator can utilize high-performance resources to reduce model run time for large cell populations, using the OpenMP[Opeb] parallel programming interface. Parallelization scalability is proven as the number of compute cores increases(along a representative range of multiprocessing), through performance tests conducted on compute-intensive cloud-computing nodes.

- Easy model and experiment configuration; the neuron model is directly input in compartmental physiological analysis terms, without the need for specialized training or programming skills. The declarative configuration format facilitates a possible future extension, allowing graphical model input.

- Support for integration with existing workflows; Cell, net and experiment models are all described through the industry-standard, both machine- and human-readable, JSON format. The simulator can run as an independent program on the user's machine, or as a containerized microservice providing a WebSockets API, so it can readily interoperate with other parts in a research process. As a proof of concept, the simulator microservice was designed to be integrated with the BrainFrame project, providing an alternate simulation backend.

# Chapter 4

# Proposed Solution

In this chapter, the software developed in the scope of this thesis is presented and the technical details and decisions regarding the implementation are discussed. The central part of the software is a neural net simulator. Based on a non-parallel prototype developed in Neurasmus B.V. additions have been made to produce a new physiologically accurate, general purpose biological neural net simulator, which provides a user-friendly modelling language and utilizes multiprocessor platforms efficiently through code parallelism. Additional components allow the simulator to run as a cloud-native application, and as a component in the BrainFrame platform developed by MicroLab. The simulator will referred to as ParModHH from now on, for brevity and clarity.

## 4.1 Neuron electrophysiological compartmental model

The ParModHH simulator assumes the compartmental level of neuron modelling. This level assumes that neural tissue consists of individual cells, and these cells are made up of distinct anatomical features(called compartments), across which the cell matter varies little. Some anatomical features may be lumped into single model compartments, as long as simulation results remain accurate, to simplify analysis. The dynamics of the internal state of each compartment are then defined directly through mathematical equations. In the case of coarse or lumped compartments, the equations are extracted through directly fitting their parameters to data, while in finer decompositions, the tubular parts of neurons are simulated through the cable equation[Tra+91].

In specific, neuron compartments are described following the extended Hodgkin-Huxley model[Lew66]. Each compartment is made up of intracellular space that contains neuroactive substances affecting its behaviour, in various concentrations, and cell membrane that contains protein complexes that selectively allow certain types of ions to move in and out of the cell, under certain conditions.

Macroscopically, each collection of such similarly-behaving complexes in a compartment or whole neuron is collectively named as an ion channel. Each ion channel complex is modelled as a chain of specific components, so that all components must be activated for ions to

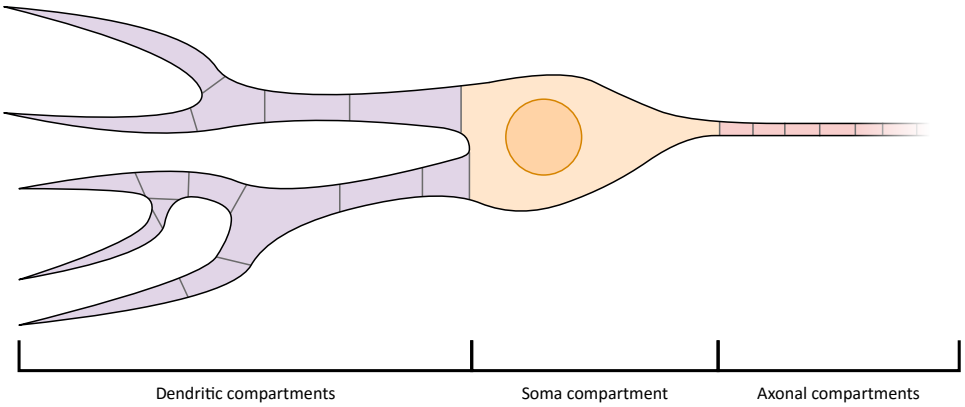



Figure 4.1: A compartmental neuron decomposition example. A fine segmentation is denoted by the thin lines inside the neuron body, while the coarser parts of the neuron are coloured differently.

pass through the channel. The probability each ion channel component is activated or deactivated may depend on membrane voltage, the local concentration of particular ions and the population of identical components that are currently active. Macroscopically, each distinct component type of an ion channel is called a gate of the ion channel, and its percentage of active components is called a gate variable. Since all components must be active for a channel complex to be open, and the probabilities each component is activated are independent, the channel's total ion transfer rate is proportional to each participating gate variable multiplied. Since a channel may contain two or more instances of the same component type, ion transfer rate may be proportional to the respective power of the corresponding gate variable.

Aside from active ion channel functionality, the electro-chemical state of cell compartments is affected by various sources. Electrical charge and chemical substances diffuse across the cells, travelling between compartments. The membrane contains electrical connections and selective ion channels with outside the cell, transferring electrical charge and specific ions in both directions. Interaction with other neurons occurs in the form of electrical synapses, where electrons directly travel across the membranes, and chemical synapses, where the release of neurotransmitters from the presynaptic neuron temporarily opens the post synaptic membrane's ion channels near the synapse.

Each compartment's state is modelled as a set of scalar variables; its intracellular electrical potential, the concentrations of ions and other molecules that affect the cell's dynamics, and the gate variables of each ion channel. Electrical flux through the cell's membrane passes through passive leaks, ion channels, and electrical synapses:

- Passive leaks behave equivalently to an ohmic leak between the membrane's two sides, in series with a voltage difference caused by the connection's molecular properties and polarity.
- Ion channels are modelled after a voltage difference in series with a linear, time-varying

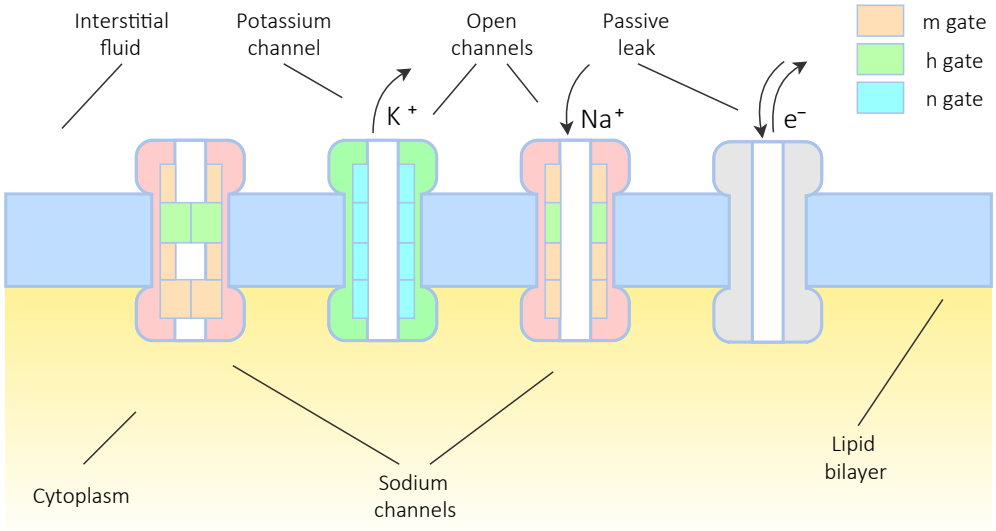



Figure 4.2: Active and passive channels on a cell membrane and the effect of gates.

resistor. Effective ion channel conductance is then a maximum conductance constant, multiplied by gate variables as previously described.

- Electrical synapse current is an antisymmetric, non-linear function of the local voltage difference between the connected cells. Since the cell membrane functions as an electrical insulator for the most part, trans-membrane current and the rate of membrane potential change are linked through each compartment's effective capacitance.

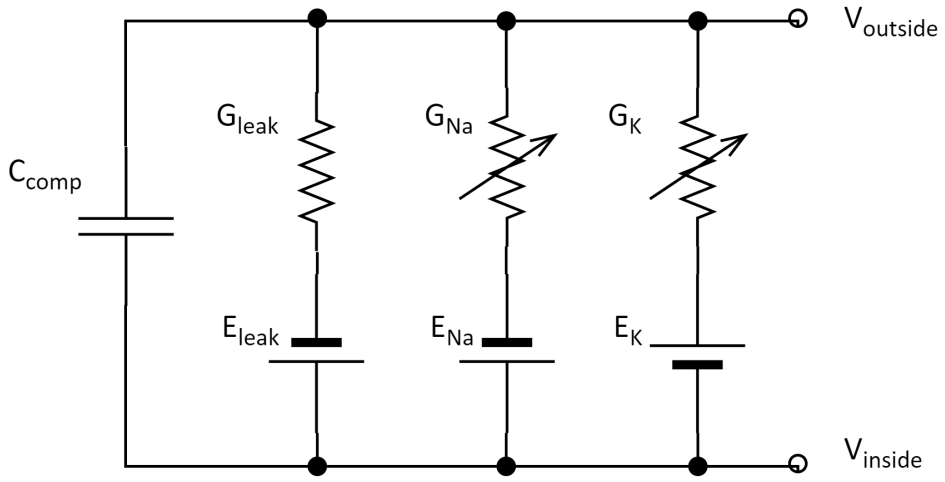



Figure 4.3: The electrical circuit equivalent of the classic Hodgkin-Huxley model.

## 4.2 Proposed simulation software

The ParModHH simulator is implemented as a non-interactive program, reading input configuration files and producing data files containing the results of the specified simulation. Input files provide the model of the neural net to be simulated, the model's initial state, the model of stimulus to be applied to the net during the simulation, experiment duration and timestep used for simulation, and auxiliary simulator configuration such as input and output file redirection, performance hints for the simulator core, etc. Output files contain the model's state variables for each simulated timestep, and meta-information about the simulation, such as run time and memory usage.

### 4.2.1 Experiment setup model

The experiment under simulation is defined by the neurons present and their initial state, the synapses connecting the neurons to each other, and the stimulation each neuron receives externally. All parameters of the model are defined through human- and machine- readable JSON[Jso] files, whose structure follows the well known to field experts Hodgkin-Huxley formulation. Each part is modelled as follows:

#### 4.2.1.1 Neuron model

The neural net model consists of a population of interconnected neurons. Each neuron is identified by a serial number for the scope of he simulation, that is used to differentiate between neurons in the experimental setup. The network's neurons may all share the same neuron model, or each neuron can be modelled differently. In the latter case, the population is described by a list of models and the serial numbers assigned to each neuron corresponds to its position in the neuron model list.

Each neuron is currently modelled in the simulator as a linear chain of interconnected cell compartments, each representing a portion of the cell. For example, dendritic tubes of a similar diameter are often lumped together, flattening the dendritic tree to a cylinder chain[RAL62]. Another example is the commonly used ball-and-stick neuron model, that segments a neuron into the soma part, where most complex chemical processes take place, and the dendrites part, which roughly models the kinetics of all stimuli received and emitted by the cell. Interaction with probes or other neurons is assumed to be made through contact among dendrites. Currently, all parts of the cell containing dendrite endpoints and synapses are assumed to be lumped into the first compartment on the chain.

Each compartment is modelled through the electrotonic features connecting it with adjacent compartments and the cell's exterior. Electrical charge is modelled to spread across compartments as if they are ohmically connected, while ion diffusion across the compartments is assumed to be negligible. The membrane's passive leak channel and ion channels are modelled after conductors in series with voltage sources(also called reversal potentials). Reversal potentials and passive leak conductance are assumed to be constant through time, while each ion channel's conductance is equal to its maximum conductance, multiplied by

its gate variables, each raised to its effective power. The electrical currents crossing the membrane are modelled in current density units, and membrane capacitance and channel conductance are likewise modelled in per-area units. As a result, electrical leaks between compartments have to be scaled by the ratio of membrane areas of the two compartments, so the same total current is correctly transferred between the compartments: $I_{leak} = J_{1,leak} * Area_1 = -J_{2,leak} * Area_2$ .

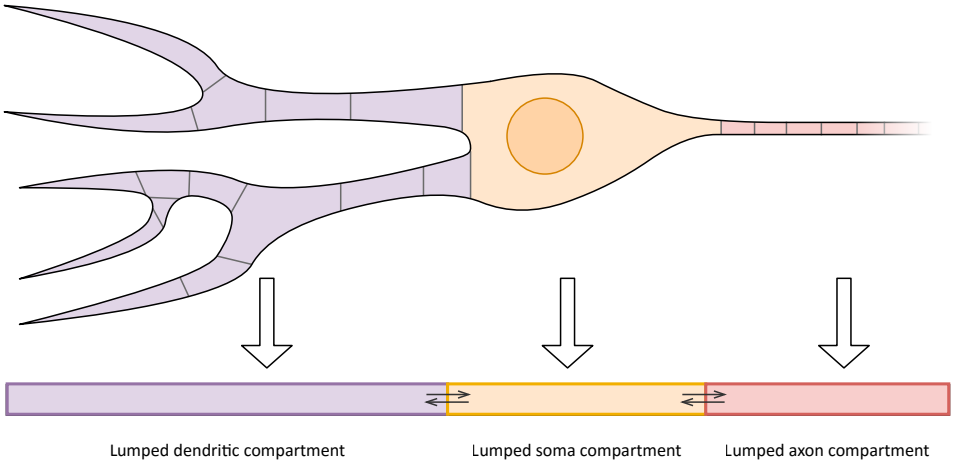



Figure 4.4: A simplified catenary neuron model, supported by the simulator.

Each ion channel gate has a set of parameters describing the behaviour of its activation variable. A gate variable may be a function of instant membrane potential (called static further on, since it displays no memory effect), or the variable's rate of change is a function of the variable's value and the compartment's membrane potential, and also calcium concentration, in calcium activated ion channels.

Gate variable dynamics are commonly described in terms of functions $\alpha$ and $\beta$, which govern the rate of the respective chemical gate structure opening and closing. In the classical Hodgkin-Huxley model, the $\alpha$ and $\beta$ values are functions of membrane potential. In the case of ion-activated ion channels, the $\alpha$ function is a function of concentration of that ion instead, and active ion concentration is modelled as a leaky bucket whose fill and leak rates are specified. An alternative dynamical behaviour considers the activation variable change rate to be proportional to the difference between the $\alpha$ function and gate variable, and inversely proportional to the $\beta$ variable. For all dynamic gates, the initial state of the variables is defined in the model.

Experiments run with uniform models can vary the initial membrane potential of the compartments of each neuron. (In the case each neuron is modelled differently, initial state is already for each cell.) For each compartment, voltage can be set to model default, or it can be a cyclic linear variation (i.e. sawtooth function) with regard to cell serial number. Variation period can be different for each compartment type.

#### 4.2.1.2 Connectivity model

The simulator supports connectivity between neurons, in the form of electrical gap junctions between lumped dendritic compartments. Gap junctions are modelled as non-linear resistors, whose conductance is a base parameter(also called weight) multiplied by a symmetric function of voltage difference(in mV) between the compartments: $g(\Delta V) = g_0 \cdot (0.8e^{-0.01 \cdot \Delta V^2} + 0.2)$ . Since the compartments containing the synapses are lumped and gap junctions work symmetrically, gap junction connectivity between each pair of cells can be described as a symmetric connectivity matrix W, with each element representing total gap junction base conductance between each pair of cells. (No connectivity between a pair of cells is equivalent to zero base conductivity.)

The simulator internally supports any arbitrary connectivity matrix, but for file size technical reasons, three implicit connectivity models are currently supported: the null connectivity model, where each neuron is separate from the others, the all-to-all uniform connectivity model, where every single pair of neurons is connected by the same gap junction weight, and the fixed probability binary connectivity model, where each possible connection between a pair of cells has the same stochastic probability of existing, and all existing pair connections have the same weight (formally put, all elements of W are Bernoulli random variables of specified magnitude and probability). Stochastic models are evaluated using a random number generator, whose seed can be specified so experiments are reproducible within the simulator version.

#### 4.2.1.3 Stimulus model

The simulator supports applying external stimuli to neurons in the simulated net, in the form of immediate electrical current injection. Current is assumed to be injected to the first compartment of each neuron. Currently, the only stimulus type supported is DC current pulse stimulus, that has a specified time of onset and duration, and may vary in intensity among cells. Like the voltage initialization option, pulse intensity can be a sawtooth function of neuron serial number.

In conjunction with an unconnected cell population (null connectivity model) and varying compartment voltage initialization, by selecting the proper repetition periods, an entire parameter combination grid of single cell experiments can be performed in a single simulation run. For example, by selecting null connectivity, initial voltage variation periods of 2 and 3 in an 2-compartment model and a variation period of 5 for stimulus intensity, 30 cells can be instantiated with the same model, with voltage initialization out of a grid of 2 different initial voltages for the first compartment and 3 different initial voltages for the second compartment, all under 5 different stimulus intensities. This technique has the limitation that grid dimensions must be co-prime; future work will support more general parameter grids.

### 4.2.2 Internal model data representation

Before simulation starts, the symbolic form used to describe the experiment model is converted to a more direct, raw binary representation that can be efficiently processed by the simulator engine.

The complex neuron modelling schema described above is converted to a more general, tabular form.

The model properties for the compartments of each neuron are stored in a matrix, where each row represents a single compartment and contains all physical parameters, along with the DC pulse stimulus parameters, for that compartment. The compartments belonging to the same neuron are represented by continuous rows, and each neuron's set of rows is added to the matrix in serial number order. The amount of compartments used for each neuron is represented through an auxiliary vector holding the cumulative compartment count per neuron.

Similarly, the properties for all gate variables present in a compartment are represented in consecutive matrix rows, grouped by ion channel. The distinction of which gates belong to which channels is resolved by extending the gate properties row with the ion channel properties, and zeroing channel properties in the rows of all gates, except for the last gate belonging to that channel. The constants defining each gate's kinetics formula are stored in the same space on the row, and a formula type integers are used to interpret these constants during simulation. Calcium-controlled ion channels are also implemented on this matrix through the use of pseudo-gate rows, which represent calcium concentration kinetics, and consecutive ion channel rows representing the calcium-activated ion channels. The amount of gates controlling each compartment are represented through an auxiliary cumulative count vector.

Finally, the system's state, comprising of compartment voltages and gate(and pseudo-gate) variables, is stored in vectors in the same order that the compartment and gate constant matrix rows are stored. As a result, static gates also have corresponding dummy state variables, which take unspecified values and should not be used.

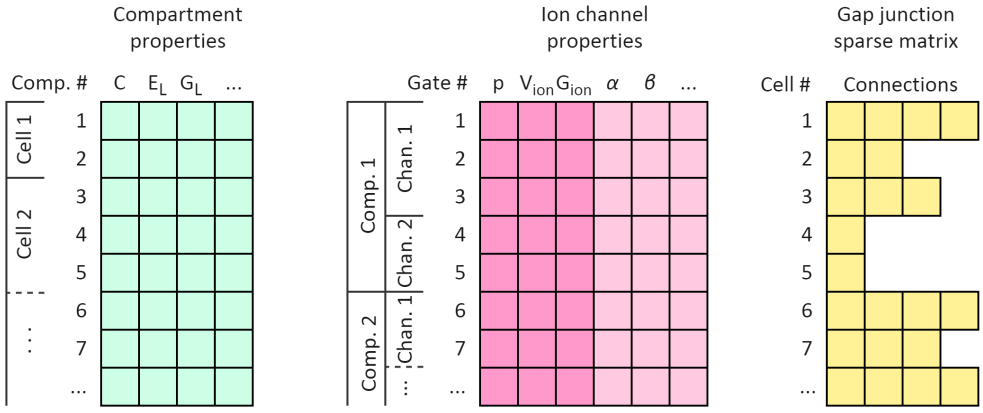



Figure 4.5: The parameter matrices internally used by the simulator.

The gap junction base conductivity between each pair of neurons can be represented as a matrix, as discussed previously. As the neural net's population increases, this matrix becomes progressively sparser. When the matrix is sparse enough, run time and memory usage can both be reduced by replacing the full matrix with a sparse representation. The simulator supports a sparse representation of the gap junction conductivity matrix, and can automatically select the most efficient representation form for each model. The sparse conductivity matrix is implemented as a list of lists. For each neuron, an associated pair of index and value lists is maintained. The index list contains the serial numbers of the neurons the associated neuron is connected to, and the value list contains the respective gap junction conductivity values.

| Cell # | | Connections | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | Target # | 3 | 4 | 6 | 8 | 13 | 25 | . . . |
| | Weight | 0.20 | 0.05 | 0.30 | 0.01 | 0.05 | 0.14 | . . . |
| 2 | Target # | 3 | 24 | | | | | |
| | Weight | 0.11 | 0.66 | | | | | |
| 3 | Target # | 1 | 2 | 7 | 42 | | | |
| | Weight | 0.20 | 0.11 | 0.08 | 0.03 | | | |
| 4 | Target # | 1 | 5 | 9 | | | | |
| | Weight | 0.05 | 0.02 | 0.09 | | | | |
| . . . | | | | | | | | |

Figure 4.6: Structure of the gap junction matrix.

### 4.2.3 Profile of a ParModHH simulation run

Initially, the program reads the JSON configuration files that describe the experiment to be simulated, whose filenames are passed through command line parameters. Configuration files describe one or more core facets of the simulation run, such as cell population, connectivity model, stimulus model, and duration of the experiment, as well as engine options such as the time delta used to approximately integrate the neuron dynamics, and performance tuning options not affecting simulation results, and meta-configuration, such as result data output options.

Configuration files can be composed, with the facets in each new configuration file added superseding previous definitions of the same facets. This functionality enables a wide variety of external tool and experimentation workflows, allowing different tools to separately export their part of the experiment to the simulator, and allows researchers to mix, match and

extend existing experiment definition component files, without the need to process and generate a new combined file.

After the configuration files are processed to form the final simulation setup, the simulator constructs the data structures used by the high-performance simulation core, such as the compartment and ion channel property, and cell connectivity matrices described previously; the parameters of each compartment are listed in the compartment properties matrix, the ion channel and gate variable parameters and kinetics formulae are decoded into the combined ion channel and gate variable property matrix, and the connectivity matrix is set to empty in the null connectivity case, full in the all-to-all connectivity case, and filled with the contents of Bernoulli trials for each possible synapse, in the fixed probability case. The amount and type of state variables in the neural net model have been determined at this point.

Since the trajectory of all state variables may not fit in memory for the whole duration of the experiment, a smaller memory buffer containing a limited amount of state variable snapshots is allocated to calculate successive steps of the simulation. Whenever this buffer fills, its contents are added to the output data file and only the previous steps required by the dynamics integration algorithm are kept for the following iterations.

Once the internal simulation data structures are constructed, the behaviour of the simulated model over time is calculated over successive discrete timesteps. The evolution of the model state throughout each timestep is approximated using ordinary differential equation integration algorithms, given the dynamics of the neural mechanisms present in the system. Currently, the Forward Euler integration algorithm is supported by the simulator. The simulator core (also called kernel) is the computation-heavy algorithm that is repeated to calculate each timestep. The computations involved are rate of change calculations, subsequent calculations to accurately predict the effect of those rates, and management of possible transient auxiliary data structures (such as the results buffer used to conserve memory in this implementation).

The factors affecting the rate of change of each state variable in the system are the currents travelling between neurons through electrical synapses, the currents diffusing across adjacent compartments in each neuron, the currents and specific ions entering and exiting each neuron compartment through ion channels and passive leaks, and the rate each ion channel gate opens and closes, which is explicitly described in the Hodgkin-Huxley formulation.

For each compartment, the cross-compartment, passive and ion channel leaks are summated using their respective model equations, to calculate total current influx and, through the membrane capacitance value, convert it to that compartment's rate of change of voltage. In the case of the special lumped dendrite compartment (or point neurons), the currents of this neuron's gap junctions to other neurons are added to total current influx. The rate membrane voltage changes on each compartment is the total current influx, divided by its membrane's capacitance. The influxes of ions interacting with specific mechanisms are also calculated, along with the current of the respective ion channels. Ion concentrations natu-

rally decay proportionally to their magnitude, adding another factor to each concentration's rate of change.

Finally, the rate of change for each gate activation variable is calculated, using its corresponding row in the gate property matrix described previously. Since the kinetics formula varies radically from gate to gate, a collection of parametric formulae is supported by the simulator. Integers in the gate property row select the type of the $\alpha$ and $\beta$ functions, their specific role in gate variable dynamics, and their dependencies on other state variables such as compartment voltage and ion concentrations. Then, the reals in the gate property row fill in the selected formulae and the final rate of change value is computed through just a few conditional blocks.

Given the current value and the rate of change for each state variable in the present timestep, their values for the next timestep are estimated. The Forward Euler algorithm adds the present value, and the rate of change multiplied by the timestep duration, to estimate the value of the next timestep. Since the values in the next timestep depend on already computed values in the present timestep, calculations for the next timestep for each state variable can take place in parallel. More sophisticated and accurate techniques, such as the Runge-Kutta methods, could be added in a future expansion.

After the previously described simulation steps are completed, the data output files are finalized by writing the remaining steps in the temporary snapshot buffer. An additional JSON file can be output, containing performance metrics such as the wall time taken to process the input files, construct the simulation core data structures, and run the simulation over the specified experiment duration, and maximum memory usage.

## 4.3 Simulation code parallelization

The ParModHH simulator was developed targeting general purpose CPUs. This computation architecture is a natural fit for the simulator since:

- General purpose CPUs are supported by all cloud and high performance computing vendors, and most vendors offer support for containerized implementations
- CPUs can simulate any type of model, while neuromorphic hardware have restrictions in model type, due to the assumptions made in their neuromorphic design
- General-purpose architectures can easily handle code variability between cells in heterogeneous networks, while vector architectures incur severe penalties in such a scenario
- The flexibility of CPUs makes running program interpreters and complex logic possible. Such features are required for a simulator that allows users to highly customize the mechanisms and structure of neuron models, without altering the simulator's internal code themselves, and with minimal delay between simulating entirely different mod-

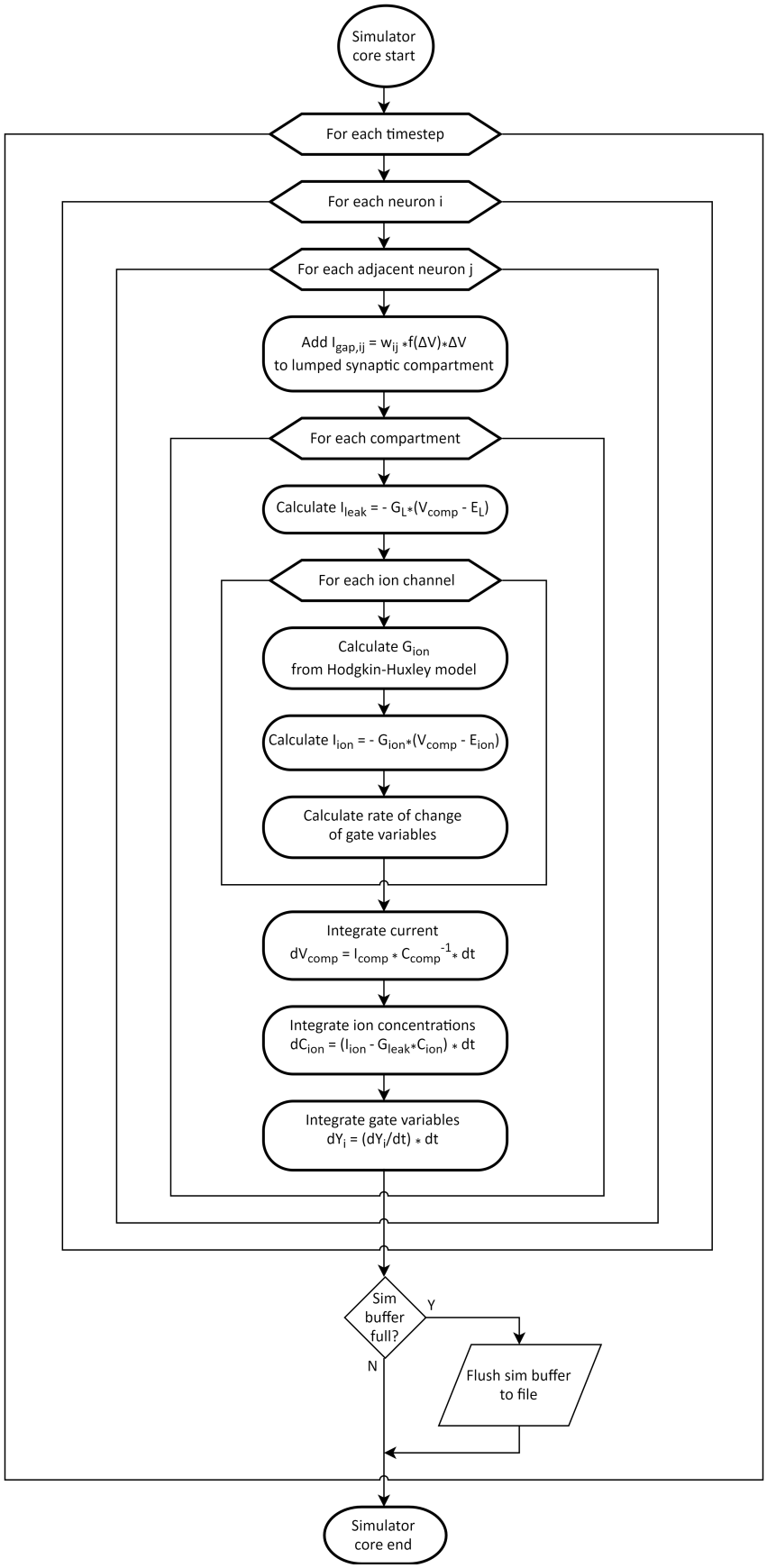



Figure 4.7: The entire simulator core flowchart.

els. Such flexible simulators enable rapid model development and evaluation, greatly empowering the research process.

The ParModHH workflow described above was enhanced with OpenMP directives, in order to utilize the parallel processing ability of multiprocessor systems. For two or more blocks of calculations to be simultaneously performed on separate processors, the calculations in each block must not depend on the other's results in any way. Each separate block of calculations assigned to be performed in parallel with others incurs computational and synchronization cost to ensure proper scheduling. Therefore, these blocks must be many enough for all processors to be busy, and few enough to not stress the OpenMP scheduler. For example, when a loop is parallelized, the OpenMP scheduler may assign whole contiguous blocks of iterations, as single tasks, to processors. This will maximize system efficiency, as long as the iterations take roughly the same time.

Due to the conductance-based, continuous-time modelling of electrical synapses, each neuron must communicate its voltage state to all the neurons it is connected to, in every single timestep. The high connectivity density of neural nets means their diameter is extremely small; typically exactly 2 for fixed-probability connectivity models supported by the simulator, and double logarithmic on average, at most logarithmic to population size for more realistic[BS09] fixed-expected-degree models[CL02]. As a result, no performance gains are expected from splitting neurons into clusters, and marking the data dependencies between clusters based on the connectivity of their neurons, in shared memory architectures. In addition, the Hodgkin-Huxley dynamics are highly non-linear and chaotic, therefore the model is unsuitable for the Parareal algorithm[LMT01] thst could make progress in multiple timesteps. Therefore, all computations per timestep are interlocked with a global synchronization step in this implementation.

Since the next values of the state variables can be computed independently in the Forward Euler integration scheme, parallelization could be applied to the level of each state variable. This approach, though, was considered problematic for multiple reasons. The subtasks resulting from a state-variable decomposition are too small and too many, while synaptic current computations for the lumped dendrite compartments, affecting compartment membrane voltage, require disproportionate computational work, presenting problems for efficient task scheduling. Aside from gap junctions, state calculations for each cell involve state variables of the same cell, so if different processors compute parts of the same cell, these variables will occupy both cache memory modules, diluting cache efficiency. In addition, the flexibility-oriented functional approach of the simulator means more complex mechanisms may be added to the model, increasing interaction between state variables and making a full decomposition of state variable calculations more convoluted. Finally, metrics gathered from conducted performance tests indicate memory traffic for gap junction computations dominates execution time for large network sizes, making the simulation performance of intracellular dynamics less important.

For these reasons, parallelization was performed across simulation of entire neurons in the net. This analysis is expected to be efficient since only the lumped dendrite compartment

voltages need to be communicated between neuron simulation tasks, the computational load for each neuron relative to task overhead is much larger than with finer approaches, yet small enough for work to be distributed evenly among processors. This parallelization approach is specified through the `#pragma omp for` directive, applied to a loop that iterates through simulation of the new state variable values for each neuron.

In case the simulation results buffer is about to fill, the operation flushing the results to the file can be performed in parallel with the new timestep. This operation is specified as an independent sibling task to simulation of all neurons, in order to mitigate possible I/O and file system delays. This property is specified through the `#pragma omp single` OpenMP directive applied to the data output statement block, the addition of the `nowait` clause to both the data output and neuron simulation loop directives, and the scoping of the `#pragma omp parallel` directive to the entire body of the timestep loop.

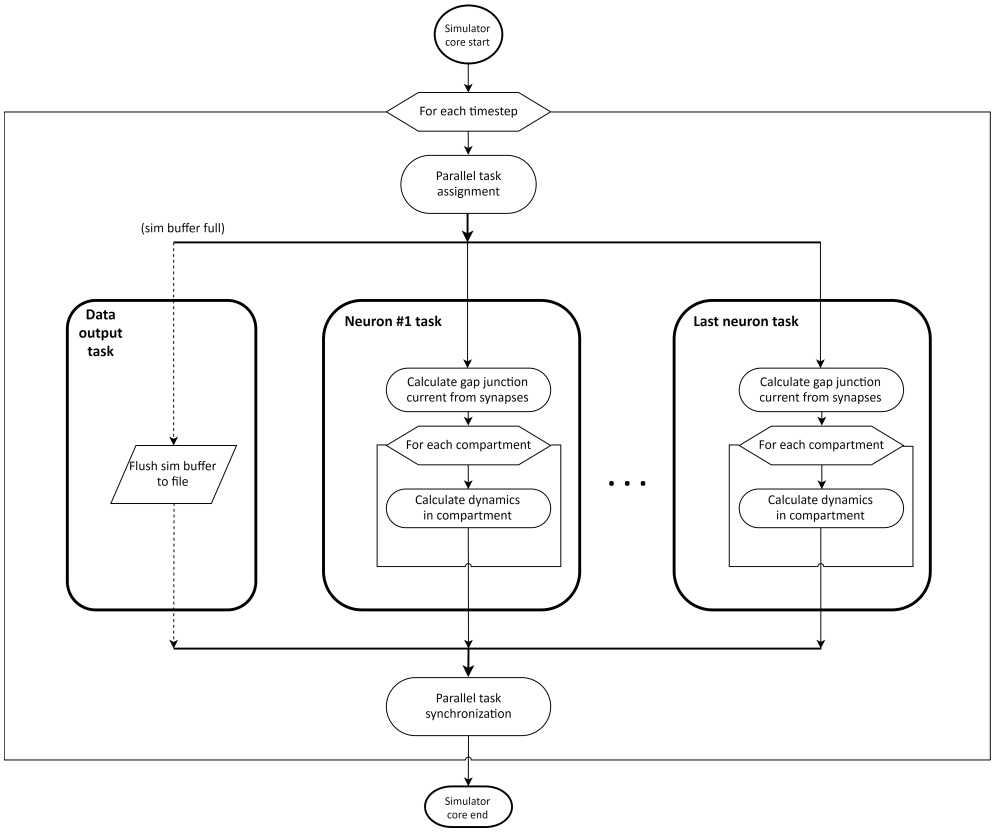



Figure 4.8: The parallelized version of the simulator core.

Performance metrics showed that for useful simulation durations, single threaded initialization takes an insignificant portion of run time. Therefore, no parallelization was applied to the file input and data structure initialization stages. Since future net connectivity models may alter this proportion, preparatory work was performed on the sparse gap junction matrix initialization algorithm; the random number generator is initialized with a different seed at the start of each row, so that successive random variable evaluations for each row can also run in parallel.

## 4.4 Cloud-native application considerations

The ParModHH simulator was initially designed as a standalone program that can be locally run on a personal computer. However, its usefulness and capabilities can be enhanced if it is deployed on a high-performance computational node, as an alternate option among other simulation backends, or integrated into a full research software solution.

### 4.4.1 Containers and the Docker architecture

A software application format that offers easy adaptation to different environments and system architectures is the application container. Originally designed to cover the needs of cloud computing, the resulting format and support infrastructure also provide the means to create fully modular software blocks, that make it possible to integrate the functionality of various software packages, without the difficulties of setting them up in a common installation.

The flexibility provided by the container format, combined with the low porting effort required to run this type application on diverse platforms and runtime environments, motivated development of a containerized version of the ParModHH simulator. The Docker platform[Doc] and container standard was selected as the development target, since it currently is the most prevalent standard in the container ecosystem, is supported by nearly all cloud-computing vendors offering container hosting, allows vertical scaling through efficient multicore execution of the ParModHH simulation software, and allows horizontal scaling through automatic orchestration of a group of simulator instances running multiple models concurrently.

The Docker platform consists of tools that handle the technicalities of container initialization, isolated execution, communication and resource use, and the orchestration of deployment and interconnection of multiple containers, which run together to implement a complex application that works under any underlying infrastructure, without special configuration by the application developer.

Containers are managed by a Docker daemon local to each host machine, through Linux namespaces and cgroups features, so that containers run the same Linux kernel as the host, each on a secure, isolated environment. The sets of file contents each container is created with are the same for containers doing the same work. These commonly used sets of contents are called images. Each image represents the software the container is supposed to run, and all images used in a project are stored in a database called a Docker image registry.

Beside spawning and managing containers, the Docker daemon can also read a description of how containers provide services and connect to each other to form a whole application, in order to manage that application. For example, it makes sure applications are responsive, by replacing the containers whose software has stopped working with new containers running the same image, and replicating containers running the same service to balance increasing load. One or more machines running the Docker daemon can also cooperate, so that the containers used by an application can be distributed among multiple

machines, further increasing deployment scalability.

### 4.4.2 Porting the simulator to the container architecture

In order to run the simulator under the Docker environment, a Docker container image must be built. An executable file is first generated from the C++ source, selecting x86-64 CPU architecture and static linkage for all libraries, in a development machine, and then added to the Docker image to be built. This executable file can run by itself with no other software additions to the Docker container. In case some libraries cannot be linked statically (e.g. OpenMP runtime for the Intel compiler), the necessary shared library files should also be added to the image, in the default shared library paths, or along with the executable by manipulating the `LD_LIBRARY_PATH` shell variable.

The Docker community typically develops script-based applications that run on an interpreted environment (such as Apache, NodeJS, Python, etc.) without specifically targeting a CPU architecture. However, since the Docker security model exposes only the host system's kernel and no system files, no portable script can be directly run on a Docker image. Instead, the shell, and other middleware that interpret scripts, must themselves be installed on the Docker image as executable binaries, adding a specific dependence to host architecture for every image built. Currently, the x86-64 architecture dominates the share of available images, and host installations. In case Docker is, for example, installed on a host with an ARM CPU, that host can only run Docker images specifically built for the ARM architecture. As a result, adding compiled binaries to Docker images may be an uncommon practice, but in itself it adds no new limitations to their execution on Docker hosts.

A way to present the ParModHH simulation functionality as a modular service to other software components is the generic isolated program execution and data output module called Communicator, developed in MicroLab. This module allows the transfer of input data to an isolated execution environment, invocation of the program to be executed, and retrieval of the files produced by the program. The whole process is mediated through a WebSockets connection, so the module can interface with software tools outside the Docker platform. The module is installed in the same container as the ParModHH simulator, and configured to use it as the execution target.

The ParModHH simulator, combined with the Communicator module, offers a modular simulator service that follows cloud application best practices[Wig], implementing a traditionally HPC workload while maintaining flexible connectivity with the other parts of the application stack and allowing quick, effortless deployment to any server infrastructure? Since it is a backend service, it does not depend to other services to function. Clients of this service are visible as inbound TCP connections, so any external tool or middleware can connect to the service seamlessly, regardless of runtime environment. All software required to build the image from a local ParModHH development machine is a set of layers of available Docker images and final ParModHH binaries, documented in a Dockerfile. The automatic image build and rollover and container instantiation process eliminate the need to modify the functionality of containers in use. The containerized implementation can be

effortlessly launched and connected to a testing application instance, that is similar to how the application runs in the production stage.

## 4.5 BrainFrame platform connectivity considerations

The BrainFrame platform is a project in active development, aiming to provide neuroscientists with an easy to use biological neural net simulation platform, incorporating various hardware acceleration architectures, in order to provide high simulation performance over a wide variety of neural nets. BrainFrame provides a web user interface that allows neuroscientists to run simulations on high-performance hardware, without the assistance of an acceleration engineer.

A core component the BrainFrame platform is the PyNN interface[BS09], receiving commonly accepted neural experiment definitions and implementing the required glue logic to invoke different simulator packages on these models. BrainFrame implements an alternative PyNN simulator backend plugin, that connects to different simulation backends to the common interface.

BrainFrame users select the simulator package, neural net model and relevant physiological parameters to be used, and describe the experiment to be performed, through the web user interface. When the whole experiment definition is complete and submitted to the system for processing, it is input into the PyNN system, which can run common model types in various simulators. Each supported simulator package can be used through the corresponding PyNN backend plugin. The BrainFrame platform includes a PyNN plugin that analyses the specified neural net model's structure and offloads computation to the most efficientacceleration hardware platform for the given net.

### 4.5.1 Interface between ParModHH simulator and BrainFrame platform

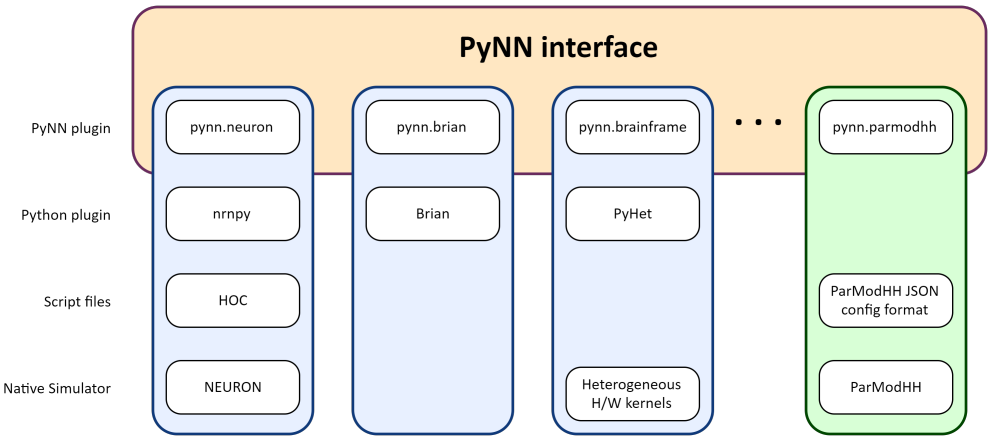



Figure 4.9: The proposed ParModHH plugin, in the context of the PyNN architecture.

The ParModHH simulator can be added to the existing BrainFrame software stack as an additional PyNN module, providing flexible ion gate-level model simulation capabilities. Since this level of neuron detail is at the time not supported through the generic PyNN cell types, the neuron models native to this simulator can be defined through the ModularHH, Compartment, IonChannel and Gate classes of the module.

The ModularHH class is a specialized subclass of PyNN multicompartmental neuron model base class, and the neuron's structure is composed from the other component classes. The population, connectivity and synapse model objects are extended from the existing base PyNN counterparts, implementing the standard ElectricalSynapse, FixedProbabilityConnector, AllToAllConnector objects of the common interface, and the Classical HH standard neuron model object by expressing it as a ParModHH model.

After the full experiment model is constructed, simulation can commence. The plugin generates JSON configuration files which describe the simulation setup to the ParModHH simulator in a temporary directory, and invokes the simulator executable to run on the generated files. The result files are then collected by the BrainFrame backend. In a future extension, the Recorder class will be properly extended so the result data are directly available from the PyNN plugin.

# Chapter 5

# Results

## 5.1 Experimental setup

An important design requirement for the ParModHH simulator was the effective simulation of large, irregular neural net models. For simulation to be effective, run time has to be minimized through clever algorithms, as well as optimal utilization of computational resources. Modern scientific computing hardware focuses on multi-processor systems and code and data parallelism to achieve high computational performance. Therefore, a set of runtime performance tests was run on multi-processor, virtual machine instances, allocated from the Amazon Elastic Cloud infrastructure[Bar18].

Allocations were made on the `c5` tier of instances. The specific tier guarantees the class of CPU architecture physically running the virtual machine. The allocation sizes used were `c5.xlarge`, `c5.2xlarge`, `c5.4xlarge`, `c5.9xlarge`, `c5.18xlarge` instances, each with 4, 8, 16, 36, 72 available vCPUs respectively. The physical CPUs used for the simulations were Intel Xeon Platinum 8124M @ 3.0 GHz (18 cores, 36 threads each). It should be noted that these CPUs have HyperThreading technology enabled, so each vCPU corresponds to a processor hyperthread. All simulations were run to utilize all available vCPUs on each instance.

In all cases, performance was measured through the wall time required to run 10,000 simulation steps of the simulation. (Preliminary tests showed that data structure initialization had an insignificant impact on total run time.) Data output was disabled, and the internal matrix representation was set to sparse in all experiments, so that algorithmic performance could be directly compared. In order to investigate neural structure processing performance, synthetic truncated models were generated for the given compartments/neuron and ion channel gates/compartment parameters. A simulation was run and timed for each combination of the vCPU count, neuron population count, net density, compartments per neuron, gates per neuron parameters specified below:

- The simulation timestep was selected at 10 microseconds, to ensure model stability.
- Simulation time was 100 milliseconds (10,000 simulation steps) for all runs.
- The vCPU count is: 4, 8, 16, 36, 72 vCPUs running in parallel.

- Neuron counts considered are: 1000, 2000, 4000, 8000, 16000 neurons.
- Neuron connectivity densities considered are: 0%, 25%, 50%, 100% of total possible synapses.
- Neuron compartment counts considered are: 1, 2, 4, 8, 16 compartments per neuron.
- Compartment gate counts considered are: 1, 2, 4, 8 gates per compartment.

In total, 2000 simulation runs were performed to cover the explored parameter grid.

## 5.2 Results presentation and discussion

For brevity, and since the simulator's core use caseis large population sizes and increasing complexity of intra-neuron modelling, only the experimental data for runs modelling 16 compartments per neuron and 8 gates per compartment (total 128 gates and 144 state variables per neuron) are displayed below:

Table 5.1: Run time for 4 vCPUs, in seconds.

| Net Density | Neuron population size | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 24.0 | 47.8 | 95.5 | 190.5 | 380.6 |
| 25% | 30.5 | 74.0 | 198.0 | 684.5 | 3296.6 |
| 50% | 36.4 | 96.6 | 292.8 | 994.1 | 5292.3 |
| 100% | 46.5 | 137.9 | 458.2 | 1567.3 | 8822.3 |

Table 5.2: Run time for 8 vCPUs, in seconds.

| Net Density | Neuron population size | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 11.8 | 23.8 | 47.4 | 95.4 | 191.3 |
| 25% | 14.9 | 35.9 | 95.5 | 349.3 | 1695.8 |
| 50% | 17.7 | 47.8 | 141.2 | 507.0 | 2682.0 |
| 100% | 22.7 | 67.2 | 219.9 | 786.0 | 4432.8 |

Performance tests were designed to investigate the simulator's strong scaling performance, as well as the effect of neuron and network complexity on performance.

Strong scaling is defined by the increase in performance as an increasing population of processors works in parallel to solve a problem instance with a fixed computational load. Strong scaling tests expose the algorithm's capability to distribute the problem's computational load across cooperating processors, minimizing processor idle time and redundant computations. Idle time may present through the natural data dependencies of the algorithm, or saturation of the processor interconnection fabric's channels. In addition, some

Table 5.3: Run time for 16 vCPUs, in seconds.

| Net Density | Neuron population size | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 6.0 | 12.3 | 24.6 | 48.0 | 95.9 |
| 25% | 7.6 | 18.7 | 49.5 | 173.9 | 831.0 |
| 50% | 9.2 | 24.6 | 72.6 | 246.4 | 1317.8 |
| 100% | 11.8 | 35.3 | 114.2 | 393.0 | 2212.7 |

Table 5.4: Run time for 36 vCPUs, in seconds.

| Net Density | Neuron population size | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 2.7 | 5.4 | 11.2 | 22.2 | 44.2 |
| 25% | 3.4 | 8.5 | 22.3 | 82.8 | 389.7 |
| 50% | 4.1 | 11.4 | 32.4 | 117.1 | 617.5 |
| 100% | 5.3 | 15.9 | 50.6 | 182.4 | 1006.7 |

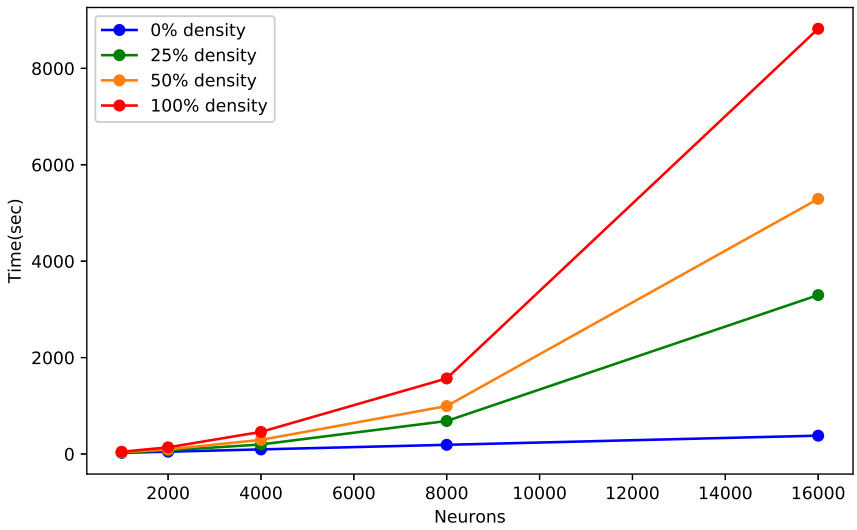



Figure 5.1: Run time for 4 vCPUs, as a function of population size.

computations may need to be replicated across processors (for example, boundary conditions and time-varying parameters).

An alternative metric in use is weak scaling, where the problem size remains fixed per processor, performance is measured by total problem size and the performance challenge is presented by the stress the total concurrent computation rate applies to the infrastructure.

It is clear from the data that run time increases linearly with connection density and

Table 5.5: Run time for 72 vCPUs, in seconds.

| Net Density | Neuron population size | | | | |
|---|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 | 16000 |
| 0% | 1.5 | 2.9 | 5.7 | 11.9 | 25.9 |
| 25% | 2.4 | 4.4 | 11.9 | 42.4 | 195.8 |
| 50% | 3.1 | 6.2 | 17.6 | 57.8 | 300.7 |
| 100% | 3.2 | 8.1 | 26.4 | 94.1 | 509.4 |

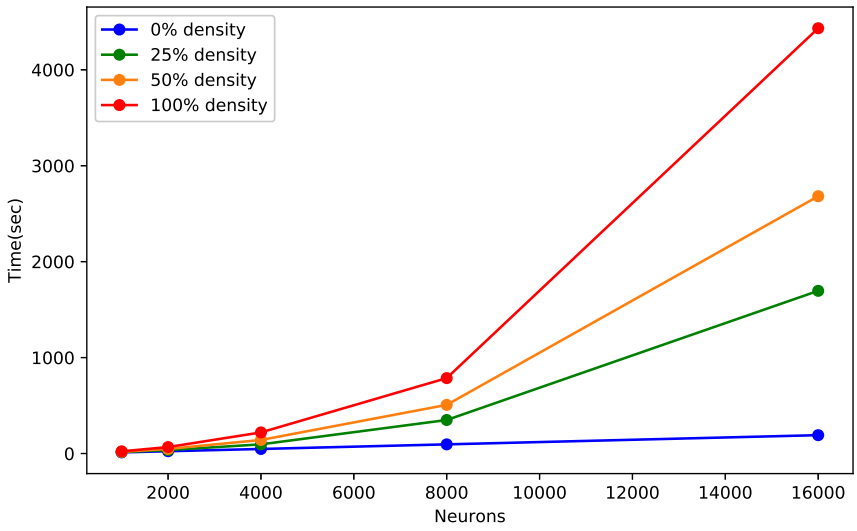



Figure 5.2: Run time for 8 vCPUs, as a function of population size.

quadratically with population size, as would be predicted by a naïve analysis of the simulation algorithm. Under this type of analysis, which is also used in computational complexity and worst-case analysis, any fundamental calculation step takes constant time, regardless of the code path taken, cache misses, out-of-execution functional unit utilization, or other instruction execution context.

For a given population size $N$, the number of possible synapses is $N\frac{N\cdot(N-1)}{2} = O(N^2)$. The number of realized synapses, under the fixed probability model, is a fraction of total possible synapses. Computations for each neuron's internal dynamics need to be performed for each neuron independently from each other's state. Internal neuron complexity is given by the model definition and, for phenomomenological compartment decomposition, is independent from neural net size. Consequently computational load for internal neuron dynamics calculations scales with $O(N)$ for increasing net size N. Therefore the total computational load scales with $O(N^2)$ for large neuron populations.

For an unconnected population, run time was measured to be linear to population size. That is also consistent with the type of analysis mentioned above, since in that case there

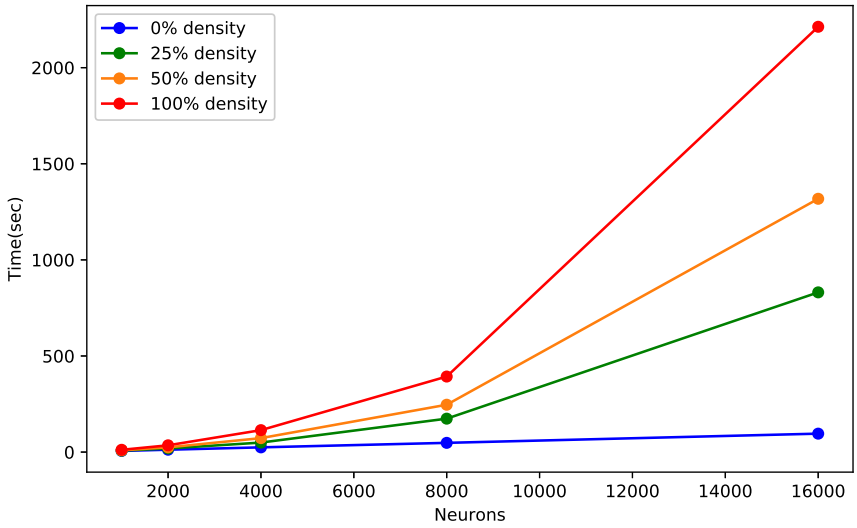



Figure 5.3: Run time for 16 vCPUs, as a function of population size.

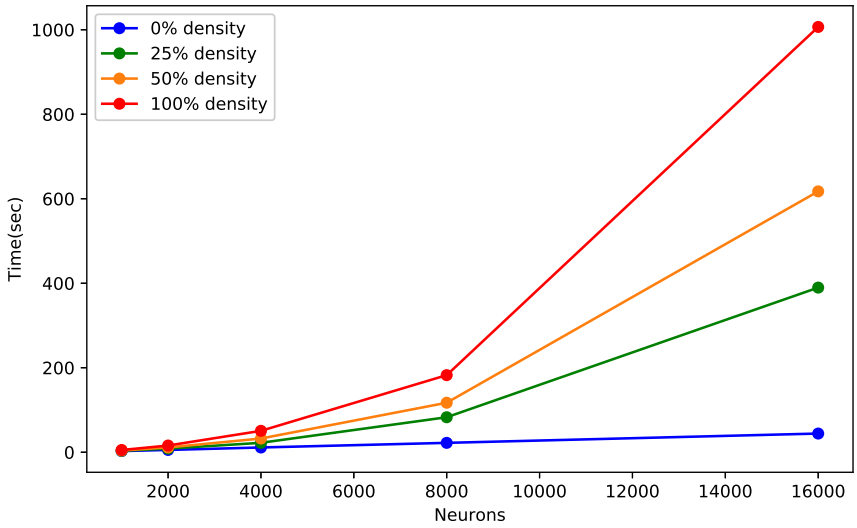



Figure 5.4: Run time for 36 vCPUs, as a function of population size.

are zero synapses, and computation is performed regarding each neuron's internal dynamics only.

For larger population sizes, measured run time is inversely proportional to the number of vCPUs used, modulo a very small constant factor (typically less than 50 seconds, no correlation to model size or complexity). This means the simulator displays ideal strong scalability, up to the maximum problem size and number of processors tested.

Ideal scalability under the per-neuron parallelization implemented in the ParModHH

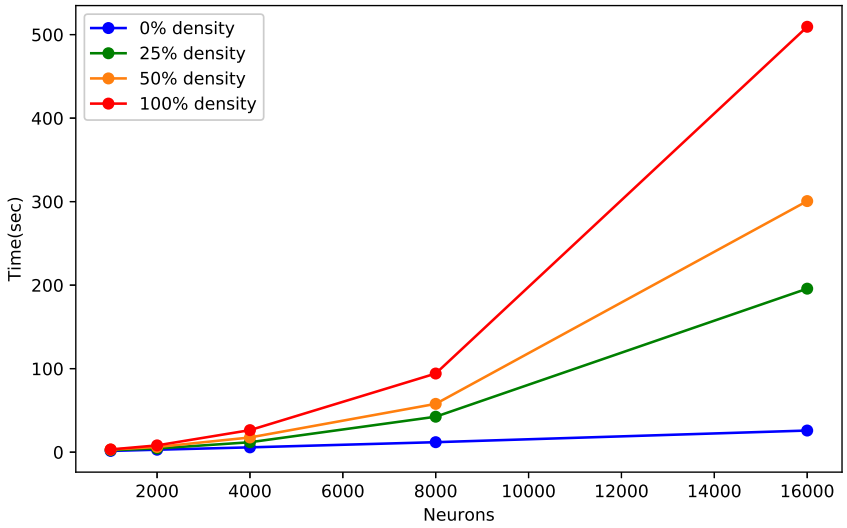



Figure 5.5: Run time for 72 vCPUs, as a function of population size.

simulator requires that for each simulation timestep, the computational load of neurons is evenly distributed among the processors, so that all processors finish their assigned work at the same time and no time is wasted in processor idling. The even distribution of work per neuron is an effect of the fixed probability neuron model, which statistically ensures all neurons have the same connectivity degree due to the law of large numbers. In addition, the delay between start of a timestep and start of parallel processing (due to per-timestep result buffer management and OpenMP parallel section initiation) and the delay of the synchronization process have to be negligible.

Beside parallel distribution of work, another potential source of speedup is the distribution of working memory. Processing for each neuron refers to specific slots in the state variables vector and a specific row in the post-synaptic connectivity matrix. Thus a smaller set of neurons per processor corresponds to a smaller amount of memory each processor needs to access exclusively, and those data may reside in a smaller, faster cache hierarchy level as a result.

Figure 4.1 shows how parallelization on a problem with maximum net population size scales ideally across the multiprocessing degrees tested.

Analysis of the experimental results showed that the relative effect of additional intra-neuron complexity on run time is an additive factor that is linear on the number of state variables added per cell. This is expected, since currently each variable depends on at most one other state variable for dynamics computations. Total run time remains inversely proportional to number of processors used. This is also expected, since the previous figures already show perfect distribution of per-neuron load, and no additional neuron interactions exist in the intra-neuron mechanisms. Figure 4.1 shows this linear scaling, for a representative neural net with maximum number of gates per compartment and varying the number

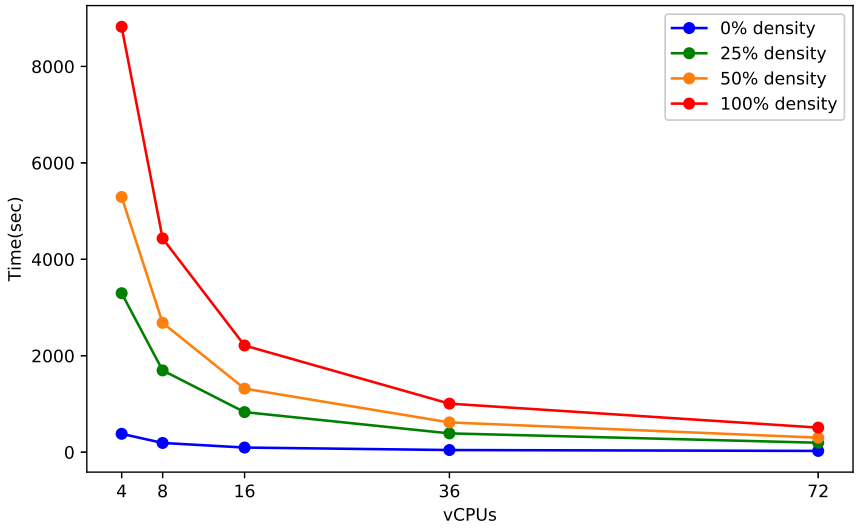



Figure 5.6: Run time for a population size of 16000 neurons.

of compartments per neuron and vCPus in use.

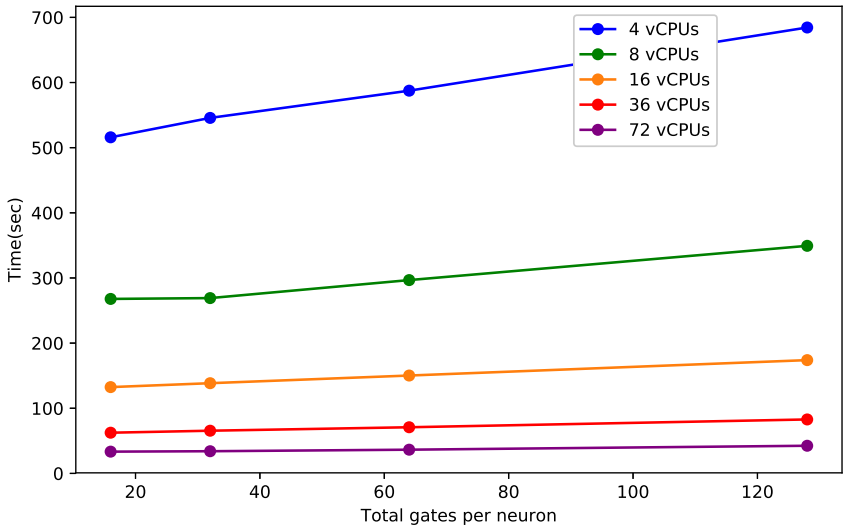



Figure 5.7: Run time for 8000 neurons and 25% density, over total amounts of gates, for 8 gates per compartment.

# Chapter 6

# Conclusion

## 6.1 Remarks

The aim of this thesis is to develop a biological neural net simulator supporting a wide range of neural models and large neural net population sizes, capable of utilizing existing high-performance and cloud computing infrastructure. The result is the ParModHH simulator, which is can simulate individually-modelled neurons under the multi-compartmental extended Hodgkin-Huxley model, supporting a wide array of ion channel models.

The simulator expresses neuron models in structural terms neuroscientists are familiar with, allowing them to directly describe a new ion channel, gate or activation condition, instead of having to learn a complex specialized programming language and deal with system administration technicalities (like when using, for example, Neuron NMODL files). As a proof of concept, a modern model for Inferior Olive neurons[DG+12] was captured into the simulator's model format.

This simulator was run on Amazon EC2 compute nodes, which belong to the higher end of consumer hardware performance and whose scale is similar single nodes in current HPC setups. It presented excellent strong scalability across the entire parameter space of population size, network density and single-neuron complexity. The simulator was deployed on the EC2 instances as a standalone container, accessed through the WebSockets interface by a local client to run the performance experiments. The simulator was also packaged into a PyNN plugin option for use by the BrainFrame simulation platform.

In conclusion, the ParModHH simulator achieves the initial model quality, high performance and deployment flexibility targets, and the underlying system can easily expand to include new capabilities and runtime platforms.

## 6.2 Future work

The ParModHH simulator, in its present state, can be directly extended in three directions: towards an extended gamut of neuronal and synaptic models, toward acceleration on alternative computational systems, and toward complete product functionality.

The neuron model's compartment connectivity model could be expanded from a catenary topology to a general graph, to include different pre- and post-synaptic compartments and self-synapses. The calcium- activated ion channel model could be generalized to a ligand-activated ion channel model, enabling physiological chemical synapse models and more ion channel models. A further generalization could even include the ion channels activated by external stimuli such as light, mechanical stress[RSP15], and pH[KW06].

The simulator kernel could be ported to different processor architectures such as GPUs and reconfigurable logic. The single-kernel requirements of GPU architectures could be mitigated through clever use of conditional instructions, by padding internal neuron data structures to equalize memory stride, and by dynamic construction of GPU compute kernels. Reconfigurable architectures could support the neuron-internal computations with an arithmetic unit that matches the simulator's model format, and with a high-bandwidth core parallelizing the processing of in-memory synaptic data.

The existing simulator could also be enhanced with features making it ready for production use, such as live simulation results streaming, simulation snapshotting for robustness, and a graphical interface for researchers to design neuron models in.

# Bibliography


[HH52] A. L. HODGKIN and A. F. HUXLEY. "Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo". In: *J. Physiol. (Lond.)* 116.4 (1952), pp. 449–472.

[RAL62] W. RALL. "Electrophysiology of a dendritic neuron model". In: *Biophys. J.* 2.2 Pt 2 (1962), pp. 145–167.

[Lew66] E.R. Lewis. "Neuroelectric potentials derived from an extended version of the Hodgkin-Huxley model". In: *Journal of Theoretical Biology* 10.1 (1966), pp. 125 –158. ISSN: 0022-5193. DOI: https://doi.org/10.1016/0022-5193(66)90181-0. URL: http://www.sciencedirect.com/science/article/pii/0022519366901810.

[Bac78] John Backus. "Can Programming Be Liberated from the Von Neumann Style?: A Functional Style and Its Algebra of Programs". In: *Commun. ACM* 21.8 (Aug. 1978), pp. 613–641. ISSN: 0001-0782. DOI: 10.1145/359576.359579. URL: http://doi.acm.org/10.1145/359576.359579.

[Mea90] C. Mead. "Neuromorphic electronic systems". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1629–1636. ISSN: 0018-9219. DOI: 10.1109/5.58356.

[Tra+91] R. D. Traub et al. "A model of a CA3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances". In: *J. Neurophysiol.* 66.2 (1991), pp. 635–650.

[Des+98] A. Destexhe et al. "Dendritic low-threshold calcium currents in thalamic relay cells". In: *J. Neurosci.* 18.10 (1998), pp. 3574–3588.

[Bas00] James B Bassingthwaighte. "Strategies for the physiome project". In: *Annals of biomedical engineering* 28.8 (2000), pp. 1043–1058.

[LMT01] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. "Résolution d'EDP par un schéma en temps «pararéel »". In: *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics* 332.7 (2001), pp. 661 –668. ISSN: 0764-4442. DOI: https://doi.org/10.1016/S0764-4442(00)01793-6. URL: http://www.sciencedirect.com/science/article/pii/S0764444200017936.

[CL02] Fan Chung and Linyuan Lu. "The average distances in random graphs with given expected degrees". In: *Proceedings of the National Academy of Sciences* 99.25 (2002), pp. 15879–15882. ISSN: 0027-8424. DOI: 10.1073/pnas.252631999. eprint: http://www.pnas.org/content/99/25/15879.full.pdf. URL: http://www.pnas.org/content/99/25/15879.

[Vog+02] R. Jacob Vogelstein et al. “Spike Timing-dependent Plasticity in the Address Domain”. In: *Proceedings of the 15th International Conference on Neural Information Processing Systems*. NIPS'02. Cambridge, MA, USA: MIT Press, 2002, pp. 1171–1178. URL: http://dl.acm.org/citation.cfm?id=2968618.2968764.

[Bal+04] G. T. Balls et al. “A large scale Monte Carlo simulator for cellular microphysiology”. In: *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* 2004, pp. 42–. DOI: 10.1109/IPDPS.2004.1302957.

[Gag04] Fabrizio Gagliardi. “The EGEE European grid infrastructure project”. In: *International Conference on High Performance Computing for Computational Science*. Springer. 2004, pp. 194–203.

[Hin+04] Michael L Hines et al. “ModelDB: a database to support computational neuroscience”. In: *Journal of computational neuroscience* 17.1 (2004), pp. 7–11.

[Aro05] Alex M. Aronov. “Predictive in silico modeling for hERG channel blockers”. In: *Drug Discovery Today* 10.2 (2005), pp. 149 –155. ISSN: 1359-6446. DOI: https://doi.org/10.1016/S1359-6446(04)03278-7. URL: http://www.sciencedirect.com/science/article/pii/S1359644604032787.

[FGH06] E. Farquhar, C. Gordon, and P. Hasler. “A field programmable neural array”. In: *2006 IEEE International Symposium on Circuits and Systems*. 2006, 4 pp.–4117. DOI: 10.1109/ISCAS.2006.1693534.

[KW06] Michaela Kress and Rainer Waldmann. “Chapter 8 Acid Sensing Ionic Channels”. In: *The Nociceptive Membrane*. Vol. 57. Current Topics in Membranes. Academic Press, 2006, pp. 241 –276. DOI: https://doi.org/10.1016/S1063-5823(06)57007-3. URL: http://www.sciencedirect.com/science/article/pii/S1063582306570073.

[Mar06] Henry Markram. “The Blue Brain Project”. In: *Nature Reviews Neuroscience* 7 (2006). Perspective, 153 EP –. URL: http://dx.doi.org/10.1038/nrn1848.

[Sut+06] L. M. Sutherland et al. “Surgical simulation: a systematic review”. In: *Ann. Surg.* 243.3 (2006), pp. 291–300.

[Boh07] M. Bohr. “A 30 Year Retrospective on Dennard's MOSFET Scaling Paper”. In: *IEEE Solid-State Circuits Society Newsletter* 12.1 (2007), pp. 11–13. ISSN: 1098-4232. DOI: 10.1109/N-SSC.2007.4785534.

[HB07] Kai M. Hynna and Kwabena Boahen. “Thermodynamically Equivalent Silicon Models of Voltage-Dependent Ion Channels”. In: *Neural Computation* 19.2 (2007), pp. 327–350. DOI: 10.1162/neco.2007.19.2.327. eprint: https://doi.org/10.1162/neco.2007.19.2.327. URL: https://doi.org/10.1162/neco.2007.19.2.327.

[Jac+07] Nicolas Jacq et al. “Grid-enabled high throughput virtual screening”. In: *Distributed, High-Performance and Grid Computing in Computational Biology*. Springer, 2007, pp. 45–59.

[HMS08] M. L. Hines, H. Markram, and F. Schurmann. “Fully implicit parallel simulation of single neurons”. In: *J Comput Neurosci* 25.3 (2008), pp. 439–448.

[KZD08] C. Kleinstreuer, Z. Zhang, and J.F. Donohue. "Targeted Drug-Aerosol Delivery in the Human Respiratory System". In: *Annual Review of Biomedical Engineering* 10.1 (2008). PMID: 18412536, pp. 195–220. DOI: 10.1146/annurev.bioeng.10.061807.160544. eprint: https://doi.org/10.1146/annurev.bioeng.10.061807.160544. URL: https://doi.org/10.1146/annurev.bioeng.10.061807.160544.

[SFM08] J. Schemmel, J. Fieres, and K. Meier. "Wafer-scale integration of analog neural networks". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008, pp. 431–438. DOI: 10.1109/IJCNN.2008.4633828.

[vBa+08] Angela van Baardwijk et al. "Radiation Dose Prescription for Non–Small-Cell Lung Cancer According to Normal Tissue Dose Constraints: An In Silico Clinical Trial". In: *International Journal of Radiation Oncology*Biology*Physics* 71.4 (2008), pp. 1103 –1110. ISSN: 0360-3016. DOI: https://doi.org/10.1016/j.ijrobp.2007.11.028. URL: http://www.sciencedirect.com/science/article/pii/S036030160704597X.

[ASS09] Yuichiro Ajima, Shinji Sumimoto, and Toshiyuki Shimizu. "Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers". In: *IEEE Computer* 42.11 (2009), pp. 36–40. DOI: 10.1109/MC.2009.370. URL: https://doi.org/10.1109/MC.2009.370.

[Ana+09] R. Ananthanarayanan et al. "The cat is out of the bag: cortical simulations with 109 neurons, 1013 synapses". In: *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. 2009, pp. 1–12. DOI: 10.1145/1654059.1654124.

[BS09] E. Bullmore and O. Sporns. "Complex brain networks: graph theoretical analysis of structural and functional systems". In: *Nat. Rev. Neurosci.* 10.3 (2009), pp. 186–198.

[KHO09] Marcus Kaiser, Claus C. Hilgetag, and Arjen van Ooyen. "A Simple Rule for Axon Outgrowth and Synaptic Competition Generates Realistic Connection Lengths and Filling Fractions". In: *Cerebral Cortex* 19.12 (2009), pp. 3001–3010. DOI: 10.1093/cercor/bhp071. eprint: /oup/backfile/content_public/journal/cercor/19/12/10.1093_cercor_bhp071/2/bhp071.pdf. URL: http://dx.doi.org/10.1093/cercor/bhp071.

[MG+09] Peter Mell, Tim Grance, et al. "The NIST definition of cloud computing". In: *National institute of standards and technology* 53.6 (2009), p. 50.

[Nag+09] Jayram Moorkanikara Nageswaran et al. "A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors". In: *Neural Networks* 22.5 (2009). Advances in Neural Networks Research: IJCNN2009, pp. 791 –800. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2009.06.028. URL: http://www.sciencedirect.com/science/article/pii/S0893608009001373.

[Pat+09] S. D. Patek et al. “In silico preclinical trials: methodology and engineering guide to closed-loop control in type 1 diabetes mellitus”. In: *J Diabetes Sci Technol* 3.2 (2009), pp. 269–282.

[Suz+09] Y. Suzuki et al. “A platform for in silico modeling of physiological systems III”. In: *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2009, pp. 2803–2806. DOI: 10.1109/IEMBS.2009.5333775.

[Cho+10] S. Choi et al. “Subthreshold membrane potential oscillations in inferior olive neurons are dynamically regulated by P/Q- and T-type calcium channels: a study in mutant mice”. In: *J. Physiol. (Lond.)* 588.Pt 16 (2010), pp. 3031–3043.

[CCT11] Richard B. Colquitt, Douglas A. Colquhoun, and Robert H. Thiele. “In silico modelling of physiologic systems”. In: *Best Practice & Research Clinical Anaesthesiology* 25.4 (2011). New Approaches in Clinical Research, pp. 499 –510. ISSN: 1521-6896. DOI: https://doi.org/10.1016/j.bpa.2011.08.006. URL: http://www.sciencedirect.com/science/article/pii/S1521689611000656.

[KW11] J. Kozloski and J. Wagner. “An Ultrascalable Solution to Large-scale Neural Tissue Simulation”. In: *Front Neuroinform* 5 (2011), p. 15.

[Wan+11] M. Wang et al. “Simulation of large neuronal networks with biophysically accurate models on graphics processors”. In: *The 2011 International Joint Conference on Neural Networks*. 2011, pp. 3184–3193. DOI: 10.1109/IJCNN.2011.6033643.

[Yam+11] Tadashi Yamazaki et al. “Simulation Platform: A cloud-based online simulation environment”. In: *Neural Networks* 24.7 (2011), pp. 693 –698. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2011.06.010. URL: http://www.sciencedirect.com/science/article/pii/S0893608011001705.

[ANV12] G. An, G. Nieman, and Y. Vodovotz. “Computational and systems biology in trauma and sepsis: current state and future perspectives”. In: *Int J Burns Trauma* 2.1 (2012), pp. 1–10.

[Bro+12] Olga R. Brook et al. “Spectral CT with Metal Artifacts Reduction Software for Improvement of Tumor Visibility in the Vicinity of Gold Fiducial Markers”. In: *Radiology* 263.3 (2012). PMID: 22416251, pp. 696–705. DOI: 10.1148/radiol.12111170. eprint: https://doi.org/10.1148/radiol.12111170. URL: https://doi.org/10.1148/radiol.12111170.

[DG+12] Jornt R. De Gruijl et al. “Climbing Fiber Burst Size and Olivary Sub-threshold Oscillations in a Network Setting”. In: *PLOS Computational Biology* 8.12 (Dec. 2012), pp. 1–10. DOI: 10.1371/journal.pcbi.1002814. URL: https://doi.org/10.1371/journal.pcbi.1002814.

[SBS12] C. J. Schneider, M. Bezaire, and I. Soltesz. “Toward a full-scale computational model of the rat dentate gyrus”. In: *Front Neural Circuits* 6 (2012), p. 83.

[Car+14] Ted Carnevale et al. “The neuroscience gateway portal: high performance computing made easy”. In: *BMC neuroscience* 15.S1 (2014), P101.

[Ger14] Liesbet Geris. “Regenerative orthopaedics: in vitro, in vivo... in silico”. In: *International orthopaedics* 38.9 (2014), pp. 1771–1778.

[Kun+14] Susanne Kunkel et al. “Spiking network simulation code for petascale computers”. In: *Frontiers in Neuroinformatics* 8 (2014), p. 78. ISSN: 1662-5196. DOI: 10.3389/fninf.2014.00078. URL: https://www.frontiersin.org/article/10.3389/fninf.2014.00078.

[Lia+14] Xiangke Liao et al. “MilkyWay-2 supercomputer: system and application”. In: *Frontiers of Computer Science* 8.3 (2014), pp. 345–356.

[Man+14] C. D. Man et al. “The UVA/PADOVA Type 1 Diabetes Simulator: New Features”. In: *J Diabetes Sci Technol* 8.1 (2014), pp. 26–34.

[Art+15] Paul Arts et al. “QPACE 2 and Domain Decomposition on the KNC”. In: *The 32nd International Symposium on Lattice Field Theory*. Vol. 214. SISSA Medialab. 2015, p. 021.

[Jel+15] Leslie C. Jellen et al. “Screening and personalizing nootropic drugs and cognitive modulator regimens in silico”. In: *Frontiers in Systems Neuroscience* 9 (2015), p. 4. ISSN: 1662-5137. DOI: 10.3389/fnsys.2015.00004. URL: https://www.frontiersin.org/article/10.3389/fnsys.2015.00004.

[Jor+15] Lyric A. Jorgenson et al. “The BRAIN Initiative: developing technology to catalyse neuroscience discovery”. In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 370.1668 (2015). ISSN: 0962-8436. DOI: 10.1098/rstb.2014.0164. eprint: http://rstb.royalsocietypublishing.org/content/370/1668/20140164.full.pdf. URL: http://rstb.royalsocietypublishi
org/content/370/1668/20140164.

[PPM15] D. Poli, V. P. Pastore, and P. Massobrio. “Functional connectivity in in vitro neuronal assemblies”. In: *Front Neural Circuits* 9 (2015), p. 57.

[Pra15] Rahul Rahul Dilip Pradhan. “Software defined infrastructures: implications for technology and business strategies for competing in the era of hyper-scale computing”. PhD thesis. Massachusetts Institute of Technology, 2015.

[RSP15] S. S. Ranade, R. Syeda, and A. Patapoutian. “Mechanically Activated Ion Channels”. In: *Neuron* 87.6 (2015), pp. 1162–1179.

[Amu+16] Katrin Amunts et al. “The Human Brain Project: Creating a European Research Infrastructure to Decode the Human Brain”. In: *Neuron* 92.3 (2016), pp. 574 –581. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2016.10.046. URL: http://www.sciencedirect.com/science/article/pii/S0896627316307966.

[Bed+16] Mathieu Bedez et al. “A fully parallel in time and space algorithm for simulating the electrical activity of a neural tissue”. In: *Journal of Neuroscience Methods* 257 (2016), pp. 17 –25. ISSN: 0165-0270. DOI: https://doi.org/10.1016/j.jneumeth.2015.09.017. URL: http://www.sciencedirect.com/science/article/pii/S0165027015003507.

[Bre+16] Markus Breit et al. “Anatomically Detailed and Large-Scale Simulations Studying Synapse Loss and Synchrony Using NeuroBox”. In: *Frontiers in Neuroanatomy* 10 (2016), p. 8. ISSN: 1662-5129. DOI: 10.3389/fnana.2016.00008. URL: https://www.frontiersin.org/article/10.3389/fnana.2016.00008.

[mPo+16] Mu ming Poo et al. “China Brain Project: Basic Neuroscience, Brain Diseases, and Brain-Inspired Computing”. In: *Neuron* 92.3 (2016), pp. 591 –596. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2016.10.050. URL: http://www.sciencedirect.com/science/article/pii/S0896627316308005.

[Mor+16] Paul D Morris et al. “Computational fluid dynamics modelling in cardiovascular medicine”. In: *Heart* 102.1 (2016), pp. 18–28. ISSN: 1355-6037. DOI: 10.1136/heartjnl-2015-308044. eprint: https://heart.bmj.com/content/102/1/18.full.pdf. URL: https://heart.bmj.com/content/102/1/18.

[Sma+16] G. Smaragdos et al. “Performance analysis of accelerated biophysically-meaningful neuron simulations”. In: *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2016, pp. 1–11. DOI: 10.1109/ISPASS.2016.7482069.

[TM16] Daniele Tartarini and Elisa Mele. “Adult Stem Cell Therapies for Wound Healing: Biomaterials and Computational Models”. In: *Frontiers in Bioengineering and Biotechnology* 3 (2016), p. 206. ISSN: 2296-4185. DOI: 10.3389/fbioe.2015.00206. URL: https://www.frontiersin.org/article/10.3389/fbioe.2015.00206.

[AB17] Oluwaseun Akeju and Emery N Brown. “Neural oscillations demonstrate that general anesthesia and sedative states are neurophysiologically distinct from sleep”. In: *Current opinion in neurobiology* 44 (2017), 178—185. ISSN: 0959-4388. DOI: 10.1016/j.conb.2017.04.011. URL: http://europepmc.org/articles/PMC5520989.

[Ger+17] Lisa Gerhardt et al. “Shifter: Containers for HPC”. In: *Journal of Physics: Conference Series* 898.8 (2017), p. 082021. URL: http://stacks.iop.org/1742-6596/898/i=8/a=082021.

[KSB17] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. “Singularity: Scientific containers for mobility of compute”. In: *PLOS ONE* 12.5 (May 2017), pp. 1–20. DOI: 10.1371/journal.pone.0177459. URL: https://doi.org/10.1371/journal.pone.0177459.

[Sil+17] Caprari Silvia et al. “bams”. In: vol. 13. 2. 2017. Chap. In silico study of the structure and function of Streptococcus mutans plasmidic proteins, p. 51. DOI: 10.1515/bams-2017-0012. URL: https://www.degruyter.com/view/j/bams.2017.13.issue-2/bams-2017-0012/bams-2017-0012.xml.

[Sma+17] Georgios Smaragdos et al. “BrainFrame: a node-level heterogeneous accelerator platform for neuron simulations”. In: *Journal of Neural Engineering* 14.6 (2017), p. 066008. URL: http://stacks.iop.org/1741-2552/14/i=6/a=066008.

[SG17] G. Stamatakos and N. Graf. "Computational Horizons in Cancer (CHIC)". In: *Clinical Therapeutics* 39.8, Supplement (2017). The Proceedings of the 13th Congress of the European Association for Clinical Pharmacology and Therapeutics, e107 –e108. ISSN: 0149-2918. DOI: https://doi.org/10.1016/j.clinthera.2017.05.333. URL: http://www.sciencedirect.com/science/article/pii/S0149291817306318.

[Bar18] Jeff Barr. *Now Available – Compute-Intensive C5 Instances for Amazon EC2 — Amazon Web Services*. 2018. URL: https://aws.amazon.com/blogs/aws/now-available-compute-intensive-c5-instances-for-amazon-ec2/.

[Car+18] Aurélie Carlier et al. "In silico clinical trials for pediatric orphan diseases". In: *Scientific reports* 8.1 (2018), p. 2465.

[Doc] *Docker overview*. 2018. URL: https://docs.docker.com/v17.12/engine/docker-overview/.

[Jor+18] Jakob Jordan et al. "Extremely Scalable Spiking Neuronal Network Simulation Code: From Laptops to Exascale Computers". In: *Frontiers in Neuroinformatics* 12 (2018), p. 2. ISSN: 1662-5196. DOI: 10.3389/fninf.2018.00002. URL: https://www.frontiersin.org/article/10.3389/fninf.2018.00002.

[WTS18] Runchun M. Wang, Chetan S. Thakur, and André van Schaik. "An FPGA-Based Massively Parallel Neuromorphic Cortex Simulator". In: *Frontiers in Neuroscience* 12 (2018), p. 213. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00213. URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00213.

[Yan+18] S. Yang et al. "Real-Time Neuromorphic System for Large-Scale Conductance-Based Spiking Neural Networks". In: *IEEE Transactions on Cybernetics* (2018), pp. 1–14. ISSN: 2168-2267. DOI: 10.1109/TCYB.2018.2823730.

[Ama] *Amazon EC2 Pricing – AWS*. URL: https://aws.amazon.com/ec2/pricing/.

[Aws] *Amazon Web Services (AWS) - Cloud Computing Services*. URL: https://aws.amazon.com/.

[Cor] *CoreOS*. URL: https://coreos.com/rkt/.

[Ent] Hewlett Packard Enterprise. *HPE SGI 8600 System - Overview*. URL: https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-a00025339en_us.

[Ari] *εισαγωγή στους υπερυπολογιστές και το σύστημα ARIS*. URL: https://hpc.grnet.gr/supercomputer/.

[Opea] *Home*. URL: https://www.opencontainers.org/.

[Lin] *Infrastructure for container projects*. URL: https://linuxcontainers.org/.

[McG+] Deirdre M. McGrath et al. "Magnetic resonance elastography of the brain: An in silico study to determine the influence of cranial anatomy". In: *Magnetic Resonance in Medicine* 76.2 (), pp. 645–662. DOI: 10.1002/mrm.25881. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.25881. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.25881.

[Azua] *Microsoft Azure Cloud Computing Platform & Services.* URL: https://azure.microsoft.com/en-us/.

[Neua] *NeuroML Database.* URL: https://neuroml-db.org/.

[Neub] *Planning support for NeuroML...* URL: https://www.neuroml.org/tool_support.

[Azub] *Pricing calculator.* URL: https://azure.microsoft.com/en-us/pricing/calculator/.

[Kub] *Production-Grade Container Orchestration.* URL: https://kubernetes.io/.

[Jso] *The JavaScript Object Notation (JSON) Data Interchange Format.* URL: https://tools.ietf.org/html/rfc8259.

[Opeb] *The OpenMP API specification for parallel programming.* URL: https://www.openmp.org/.

[Opec] *What is OpenStack?* URL: https://www.openstack.org/software/.

[Wig] Adam Wiggins. *The Twelve-Factor App.* URL: https://12factor.net/.